# MỤC LỤC

Trang

TRANG PHỤ BÌA ..i

LỜI CAM ĐOAN .ii

LỜI CẢM ƠN iii

[**MỤC LỤC** 1](#_Toc402884465)

[DANH MỤC CÁC CHỮ VIẾT TẮT 4](#_Toc402884466)

[DANH MỤC CÁC BẢNG 4](#_Toc402884467)

[DANH MỤC CÁC SƠ ĐỒ 5](#_Toc402884468)

[MỞ ĐẦU 8](#_Toc402884469)

[Lý do chọn đề tài (tính cấp thiết của đề tài) 8](#_Toc402884470)

[Mục đích, mục tiêu nghiên cứu của đề tài 8](#_Toc402884471)

[Đối tượng và phạm vi nghiên cứu 8](#_Toc402884472)

[Phương pháp nghiên cứu 9](#_Toc402884473)

[Kết cấu của đề tài 9](#_Toc402884474)

[**NỘI DUNG** 11](#_Toc402884475)

[CHƯƠNG 1: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG 11](#_Toc402884476)

[1.1. Tiếp nhận và xử lý yêu cầu 11](#_Toc402884477)

[1.1.1. Yêu cầu chức năng nghiệp vụ 11](#_Toc402884478)

[1.1.2. Yêu cầu chức năng thống kê báo cáo 12](#_Toc402884479)

[1.1.3. Yêu cầu kỹ thuật 12](#_Toc402884480)

[1.2. Lược đồ quan niệm 14](#_Toc402884481)

[1.3. Lược đồ trường hợp sử dụng (use case) 16](#_Toc402884482)

[1.4. Lược đồ CSDL mức vật lý 16](#_Toc402884483)

[1.5. Lược đồ lớp (class) 18](#_Toc402884484)

[1.5.1. Lược đồ các lớp thực thể 18](#_Toc402884485)

[1.5.2. Các lớp thư viện liên quan 22](#_Toc402884486)

[1.6. Lược đồ tuần tự (sequences) 23](#_Toc402884487)

[1.6.1. Thêm thiết bị mới 24](#_Toc402884488)

[1.6.2. Xóa thiết bị khỏi hệ thống 25](#_Toc402884489)

[1.6.3. Chuyển thiết bị giữa các phòng hoặc chuyển tình trạng thiết bị 26](#_Toc402884490)

[1.6.4. Loại bỏ thiết bị khỏi phòng 27](#_Toc402884491)

[1.6.5. Chức năng chuyển tình trạng thiết bị 28](#_Toc402884492)

[CHƯƠNG 2: THỰC THI 29](#_Toc402884493)

[2.1. Môi trường lập trình và phát triển ứng dụng 29](#_Toc402884494)

[2.2. Mô hình tổ chức ứng dụng 29](#_Toc402884495)

[2.2.1. Tổ chức ứng dụng theo hướng module hóa trong lập trình đa nền tảng (cross - platform) 29](#_Toc402884496)

[2.2.2. Mô hình 3 lớp trong ứng dụng hướng dữ liệu 30](#_Toc402884497)

[2.2.3. Mô hình MVP (Model - View - Presenter) dành cho ứng dụng WinForm Desktop 31](#_Toc402884498)

[2.2.4. Mô hình ASP.NET Web Forms dành cho ứng dụng Web 32](#_Toc402884499)

[2.3. Các công nghệ và kỹ thuật lập trình được áp dụng 32](#_Toc402884500)

[2.3.1. Công nghệ Entity Framework (EF) trong lập trình dữ liệu hướng đối tượng (OOP) 32](#_Toc402884501)

[2.3.2. Công nghệ Sync Framework của Microsoft trong đồng bộ CSDL tập trung 42](#_Toc402884502)

[2.3.3. Công nghệ DevExpress trong lập trình giao diện 49](#_Toc402884503)

[2.3.4. Công nghệ giao diện tùy biến (responsive design) dành cho ứng dụng Web 58](#_Toc402884504)

[2.4. Kết quả thực thi (các màn hình chức năng) 63](#_Toc402884505)

[CHƯƠNG 3: KIỂM THỬ VÀ TRIỂN KHAI 67](#_Toc402884506)

[3.1. Kiểm thử tự động mức mã nguồn (Unit test) 67](#_Toc402884507)

[3.1.1. Kiểm thử hộp đen (Black box testing) 67](#_Toc402884508)

[3.1.2. Mô hình kiểm thử AAA (Arrange - Act - Assert) 67](#_Toc402884509)

[3.2. Kiểm thử chấp nhận (Acceptance test) 67](#_Toc402884510)

[3.3. Các mô hình triển khai 75](#_Toc402884511)

[**KẾT LUẬN VÀ KIẾN NGHỊ** 77](#_Toc402884512)

[DANH MỤC TÀI LIỆU THAM KHẢO 79](#_Toc402884513)

[PHỤ LỤC A 81](#_Toc402884514)

[PHỤ LỤC B 84](#_Toc402884515)

[PHỤ LỤC C 89](#_Toc402884516)

DANH MỤC CÁC CHỮ VIẾT TẮT

|  |  |  |
| --- | --- | --- |
| CSDL | : | Cơ sở dữ liệu |
| EF | : | Entity Framework |
| RWD | : | Responsive Web Design |
|  |  |  |

DANH MỤC CÁC BẢNG

|  |  |  |
| --- | --- | --- |
| **TT** | **Tên bảng** | **Trang** |
| 1 | Bảng 3.1: Kết quả kiểm thử chấp nhận | 68 |
| 2 | Bảng PL1.1: Sơ đồ bảng COSOS | 81 |
| 3 | Bảng PL1.2: Sơ đồ bảng VITRIS | 81 |
| 4 | Bảng PL1.3: Sơ đồ bảng PHONGS | 82 |
| 5 | Bảng PL1.4: Sơ đồ bảng THIETBIS | 82 |
| 6 | Bảng PL1.5: Sơ đồ bảng CTTHIETBIS | 82 |
| 7 | Bảng PL2.1: Thiết kế giao diện \_CRUDInterface<T> | 84 |
| 8 | Bảng PL2.2: Thiết kế giao diện \_EFEventRegisterInterface | 85 |
| 9 | Bảng PL2.3: Thiết kế lớp ảo \_EntityAbstract1<T> | 86 |
| 10 | Bảng PL2.4: Thiết kế lớp ảo \_EntityAbstract2<T> | 87 |
| 11 | Bảng PL3.1: Thiết kế lớp cứng CoSo | 89 |
| 12 | Bảng PL3.2: Thiết kế lớp cứng ViTri | 90 |
| 13 | Bảng PL3.3: Thiết kế lớp cứng Phong | 90 |
| 14 | Bảng PL3.4: Thiết kế lớp cứng ThietBi | 91 |
| 15 | Bảng PL3.5: Thiết kế lớp cứng CTThietBi | 92 |

DANH MỤC CÁC SƠ ĐỒ

|  |  |  |
| --- | --- | --- |
| **TT** | **Tên sơ đồ** | **Trang** |
| 1 | Sơ đồ 1.1: Sơ đồ quan niệm quan hệ phòng - thiết bị | 14 |
| 2 | Sơ đồ 1.2: Sơ đồ quan niệm nhân viên phụ trách phòng | 14 |
| 3 | Sơ đồ 1.3: Sơ đồ quan niệm quan hệ phân quyền | 15 |
| 4 | Sơ đồ 1.4: Sơ đồ quan niệm quan hệ hình ảnh | 15 |
| 5 | Sơ đồ 1.5: Sơ đồ quan niệm quan hệ mượn phòng | 16 |
| 6 | Sơ đồ 1.6: Sơ đồ trường hợp sử dụng (use case) | 16 |
| 7 | Sơ đồ 1.7: Sơ đồ CSDL vật lý quan hệ phòng - thiết bị | 17 |
| 8 | Sơ đồ 1.8: Sơ đồ CSDL vật lý quan hệ phòng - vị trí | 17 |
| 9 | Sơ đồ 1.9: Sơ đồ CSDL mức vật lý quan hệ phòng - sự cố - nhân viên phụ trách | 17 |
| 10 | Sơ đồ 1.10: Sơ đồ CSDL mức vật lý quan hệ phân quyền | 18 |
| 11 | Sơ đồ 1.11: Sơ đồ CSDL mức vật lý quan hệ phiếu mượn phòng | 18 |
| 12 | Sơ đồ 1.12: Sơ đồ CSDL mức vật lý quan hệ hình ảnh | 18 |
| 13 | Sơ đồ 1.13: Sơ đồ kế thừa các lớp thực thể | 19 |
| 14 | Sơ đồ 1.14: Sơ đồ lớp quan hệ phòng - thiết bị | 20 |
| 15 | Sơ đồ 1.15: Sơ đồ lớp quan hệ phòng - vị trí | 20 |
| 15 | Sơ đồ 1.16: Sơ đồ lớp quan hệ phân quyền | 21 |
| 16 | Sơ đồ 1.17: Sơ đồ lớp quan hệ hình ảnh | 21 |
| 17 | Sơ đồ 1.18: Sơ đồ lớp quan hệ phân công phòng | 21 |
| 18 | Sơ đồ 1.19: Sơ đồ lớp quan hệ mượn phòng | 21 |
| 19 | Sơ đồ 1.20: Sơ đồ tuần tự cho chức năng "Thêm thiết bị mới" | 24 |
| 20 | Sơ đồ 1.21: Sơ đồ tuần tự cho chức năng "Xóa thiết bị khỏi hệ thống" | 25 |
| 21 | Sơ đồ 1.22: Sơ đồ tuần tự cho chức năng "Chuyển thiết bị giữa các phòng hoặc chuyển tình trạng thiết bị" | 26 |
| 22 | Sơ đồ 1.23: Sơ đồ tuần tự cho chức năng "Loại bỏ thiết bị khỏi phòng" | 27 |
| 23 | Sơ đồ 1.24: Sơ đồ tuần tự cho chức năng "Chuyển tình trạng thiết bị" | 28 |
| 24 | Sơ đồ 2.1: Mô hình quan niệm tổ chức ứng dụng hướng module hóa | 29 |
| 25 | Sơ đồ 2.2: Mô hình chi tiết tổ chức ứng dụng hướng module hóa | 30 |
| 26 | Sơ đồ 2.3: Mô hình MVP | 31 |
| 27 | Sơ đồ 2.4: Mô hình triển khai TPC trong kế thừa | 34 |
| 28 | Sơ đồ 2.5: Cách hoạt động của kỹ thuật truy vấn lồng trong LINQ | 36 |
| 29 | Sơ đồ 2.6: Tham chiếu ngược trên các quan hệ 1 - n, n - n | 36 |
| 30 | Sơ đồ 2.7: Tương thích ngược phiên bản CSDL | 37 |
| 31 | Sơ đồ 2.8: Tương thích xuôi phiên bản CSDL | 38 |
| 32 | Sơ đồ 2.9: Không tương thích phiên bản CSDL (trường hợp 1) | 38 |
| 33 | Sơ đồ 2.10: Không tương thích phiên bản CSDL (trường hợp 2) | 38 |
| 34 | Sơ đồ 2.11: Giao tiếp 2 chiều trong mô hình dữ liệu hướng sự kiện | 40 |
| 35 | Sơ đồ 2.12: Cách hoạt động giữa Singleton và Database Context | 42 |
| 36 | Sơ đồ 2.13: Cách hoạt động của Sync Framework | 43 |
| 37 | Sơ đồ 2.14: Đụng độ khóa chính do sử dụng kiểu dữ liệu tự động tăng | 47 |
| 38 | Sơ đồ 2.15: Minh họa giải quyết đụng độ khi sử dụng khóa chính là GUID | 48 |
| 39 | Sơ đồ 2.16: Minh họa semaphore đơn tiến trình trong xử lý đa luồng trên hàng đợi | 49 |
| 40 | Sơ đồ 2.17: So sánh giữa kích thước trang Web và độ phân giải màn hình | 61 |
| 41 | Sơ đồ 3.1: Mô hình triển khai hệ thống phần mềm | 68 |

MỞ ĐẦU

## Lý do chọn đề tài (tính cấp thiết của đề tài)

- Để đáp ứng nhu cầu về quản lý các thiết bị cho Trường ĐH Sài Gòn, Ban Hạ tầng Cơ sở mong muốn có được một phần mềm quản lý tập trung các thiết bị hiện có ở tất cả các cơ sở, nhất là các loại thiết bị có giá thành cao và dễ hỏng hóc. Nhờ đó trường sẽ dễ dàng nắm được thông tin về tình hình thiết bị ngay khi cần.

- Yêu cầu đặt ra cho khoa Công nghệ thông tin của trường cũng như sinh viên là phải nghiên cứu, thiết kế và tạo ra được phần mềm đúng nhu cầu sử dụng của trường. Đồng thời, đây cũng là cơ hội để sinh viên có được đề tài và hoàn thành khóa luận tốt nghiệp.

- Vì vậy nhóm quyết định chọn đề tài:

"Phần mềm quản lý tài sản Trường ĐH Sài Gòn - Phân hệ quản lý thiết bị"

\*Đồng thời với khóa luận này còn có một khóa luận nữa của nhóm sinh viên Nguyễn Hoàng Thanh, Huỳnh Công Khánh làm về đề tài "Phần mềm quản lý tài sản Trường ĐH Sài Gòn - Phân hệ quản lý tài sản cố định" (cùng một người hướng dẫn), 2 nhóm đã hợp tác để làm chung một hệ thống lớn là "Phần mềm quản lý tài sản Trường ĐH Sài Gòn". Đề tài này là 1 trong 2 phân hệ chức năng chính của hệ thống lớn nói trên. Do đó, giữa 2 khóa luận có rất nhiều điểm tương đồng vì sử dụng chung các công nghệ và quy trình phát triển.

## Mục đích, mục tiêu nghiên cứu của đề tài

- Mục đích: tạo ra được ứng dụng quản lý thiết bị cho Trường ĐH Sài Gòn.

- Mục tiêu:

+ Nghiên cứu các công nghệ hiện đại có liên quan và xoay quanh đề tài, chọn ra công nghệ phù hợp.

+ Tìm hiểu các mô hình phát triển và xây dựng phần mềm tiên tiến.

+ Thu thập các số liệu thực tế phục vụ cho nghiên cứu.

+ Tin học hóa các nghiệp vụ quản lý thiết bị.

## Đối tượng và phạm vi nghiên cứu

- Đối tượng nghiên cứu:

“Nghiệp vụ và quy trình quản lý thiết bị Trường ĐH Sài Gòn”.

- Phạm vi nghiên cứu:

+ Chỉ nghiên cứu giải pháp cho Trường ĐH Sài Gòn.

+ Các số liệu thực tế chỉ được lấy từ Ban Hạ tầng Cơ sở.

+ Các công nghệ có liên quan sẽ chỉ được nghiên cứu ở mức áp dụng vào thực tiễn, không đi sâu vào các lý thuyết bên trong.

+ Chỉ nghiên cứu các công nghệ có tính ứng dụng cao, và được cập nhật gần đây không quá 10 năm.

+ Chỉ tìm hiểu và khảo sát các nhà cung cấp dịch vụ hiện có trong nước như: VDC, Mắt bão, PA Việt Nam, Nhân Hòa,...

## Phương pháp nghiên cứu

- Phương pháp tổng hợp, phân tích (thu thập tài liệu của các tác giả trong, ngoài nước có liên quan đến đề tài một cách có chọn lọc).

- Phương pháp so sánh và đưa ra quyết định (tìm ưu nhược điểm của các phương án, chọn phương án hợp lý).

- Nghiên cứu và phát triển lý thuyết xoay quanh đề tài.

- Phương pháp thử và sai.

- Phương pháp chuyên gia (tham khảo ý kiến của các chuyên gia về lĩnh vực, khía cạnh cụ thể).

- Phương pháp phản biện và tự phản biện.

## Kết cấu của đề tài

- Đề tài gồm có 3 phần: phần mở đầu, phần nội dung được trình bày theo từng chương, phần kết luận và kiến nghị.

- Phần nội dung gồm có 3 chương:

+ CHƯƠNG 1: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

=> Trình bày các yêu cầu hệ thống, các bản phân tích, sơ đồ thiết kế cần thiết để chuẩn bị cho bước thực thi. Giai đoạn này quan trọng vì sẽ ảnh hưởng đến chất lượng của ứng dụng cũng như khả năng mở rộng về sau.

+ CHƯƠNG 2: THỰC THI

=> Trình bày các công nghệ được áp dụng, và kết quả thực thi.

+ CHƯƠNG 3: KIỂM THỬ VÀ TRIỂN KHAI

=> Sau bước thực thi thì việc kiểm thử và nghiệm thu phần mềm sẽ được diễn ra. Phần triển khai thực tế phần mềm cũng sẽ được trình này ở đây.

NỘI DUNG

CHƯƠNG 1: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

## 1.1. Tiếp nhận và xử lý yêu cầu

### 1.1.1. Yêu cầu chức năng nghiệp vụ

- Trường ĐH Sài Gòn có nhiều cơ sở, mỗi cơ sở có nhiều dãy, mỗi dãy có nhiều tầng. Mỗi cơ sở phải có: tên, địa chỉ (sử dụng trong hướng dẫn chỉ đường), mô tả,...Mỗi dãy, tầng cần phải có: tên, mô tả,...

- Ban Hạ tầng Cơ sở của Trường ĐH Sài Gòn muốn quản lý các thiết bị theo từng phòng.

- Mỗi phòng chứa nhiều thiết bị, các thiết bị có thể di chuyển qua lại giữa các phòng hoặc thay đổi tình trạng của thiết bị trong phòng (đang sử dụng, bị hư, đang sửa chữa,...).

- Mỗi thiết bị cần có các thông tin: mã thiết bị, tên thiết bị, loại thiết bị, ngày sử dụng, tình trạng, hình ảnh...

- Mỗi phòng cần lưu trữ các thông tin sau: mã phòng, tên phòng, hình ảnh, vị trí (thuộc cơ sở - dãy - tầng), mô tả,...

- Ngoài các thông tin cơ bản thì mỗi phòng còn phải lưu trữ thông tin nhân viên phụ trách phòng (quản lý mở cửa, dọn dẹp vệ sinh phòng) và thông tin quản trị viên chịu trách nhiệm quản lý chung. Phải có chức năng phân công hàng loạt.

- Phòng có thể có nhiều sự cố xảy ra trong quá trình sử dụng, cần phải lưu trữ thông tin sự cố của từng phòng như: tên sự cố, tình trạng, mô tả, hình ảnh sự cố,....Có thể xem được lịch sử cập nhật và sửa đổi sự cố của một phòng.

- Với mong muốn tin học hóa, Ban Hạ tầng Cơ sở cũng yêu cầu bổ sung chức năng mượn phòng và duyệt phòng qua mạng, có gửi email xác báo. Theo đó, mỗi phiếu mượn phòng cần được kê khai như mẫu hiện hành đang sử dụng, bao gồm các thông tin như: thời gian mượn/trả, số lượng phòng cần mượn, số lượng sinh viên, đơn vị đệ trình,...Chức năng này có thể làm việc qua Web.

- Dữ liệu có thể thống kê, xuất báo cáo theo các điều kiện lọc như: ngày, tình trạng, loại thiết bị, cơ sở. Có thể kết xuất dữ liệu báo cáo ra các tập tin thông dụng như pdf, excel, html để thiết kế lại hoặc in ấn trực tiếp.

- Đảm bảo phân quyền chặt chẽ, nhất quán. Cho phép phân quyền trên các đối tượng cụ thể như: cơ sở, dãy, tầng, phòng.

- Mỗi thao tác trên cơ sở dữ liệu điều phải lưu lại vào nhật ký hệ thống để quản trị viên dễ dàng theo dõi và nắm bắt tình hình.

- Ứng dụng có thể được truy cập qua mạng toàn cầu Internet.

### 1.1.2. Yêu cầu chức năng thống kê báo cáo

- Thống kê thiết bị/số lượng thiết bị theo các điều kiện lọc: tình trạng, loại thiết bị, ngày sử dụng, ngày cập nhật cuối, vị trí (cơ sở, dãy, tầng, phòng).

- Thống kê danh sách nhân viên phụ trách hiện có toàn trường hoặc theo cơ sở.

- Thống kê danh sách phòng hiện có toàn trường hoặc theo cơ sở.

\*Các thống kê trên đều phải xuất được báo cáo để in ấn hoặc xử lý hậu kỳ.

### 1.1.3. Yêu cầu kỹ thuật

- Các yêu cầu kỹ thuật sau đây được xem xét và đưa ra dựa trên các khảo sát về cơ sở hạ tầng hiện tại của các nhà cung cấp dịch vụ liên quan và hạ tầng trang thiết bị hiện có của trường.

#### 1.1.3.1 Yêu cầu về hệ thống thông tin

- Triển khai được trên hệ thống mạng nội bộ của trường hoặc trên mạng Internet toàn cầu hiện hành.

- Tương thích và vận hành tốt trên các giao thức mạng phổ biến hiện tại như: IP (IPv4, IPv6), FTP (dùng trong tải lên/xuống các tập tin), HTTP 1.1 (dùng trong gửi nhận dữ liệu), SMTP - IMAP (dùng trong gửi và nhận email),...

- Có khả năng làm việc khi không có mạng, đồng bộ dữ liệu lên máy chủ tập trung khi có mạng.

- Đảm bảo tính nhất quán về mặt dữ liệu giữa các ứng dụng trên các nền tảng khác nhau, ví dụ: giữa ứng dụng Desktop và ứng dụng Web.

- Yêu cầu chung về bảo mật dữ liệu và an toàn thông tin.

+ Dữ liệu phải có khả năng sao lưu và dễ dàng khôi phục lại khi cần thiết.

+ Dữ liệu chỉ được truy cập bởi những người dùng có thẩm quyền.

+ Các nghiệp vụ liên quan đến bảo mật:

* Mã hóa mã nguồn, chống dịch ngược.
* Mã hóa các thông tin nhạy cảm khi lưu trữ (mật khẩu người dùng, cấu hình cài đặt,...) theo tiêu chuẩn mã hóa dữ liệu Advanced Encryption Standard (AES) do Viện Tiêu chuẩn và Công nghệ quốc gia Hoa Kỳ (National Institute Standards and Technology - NIST) phát hành ngày 26/11/2001.
* Các dữ liệu truyền tải giữa máy chủ và máy tính cá nhân người dùng cần đảm bảo độ an toàn bằng cách mã hóa trước khi truyền qua mạng.

#### 1.1.3.2. Yêu cầu phần cứng

- Vận hành tốt trên các hệ thống máy tính hiện có của trường, cấu hình hệ thống máy tính tối thiểu đề nghị:

+ CPU: xung nhịp 1.0 Ghz hoặc cao hơn.

+ RAM: dung lượng 512 MB hoặc cao hơn.

+ Đĩa cứng: dung lượng trống tối thiểu 5GB

+ Kích cỡ 15 inch hoặc lớn hơn, độ phân giải 1024 x 768 hoặc cao hơn.

#### 1.1.3.3. Yêu cầu phần mềm

- Ứng dụng chạy trên máy tính cá nhân PC (Windows Desktop Application)

+ Hệ điều hành Windows 7 hoặc mới hơn.

+ Tương thích .NET Framework 4.0 hoặc mới hơn. Phiên bản .NET bản đầy đủ (Full) (vì mỗi phiên bản .NET nhiều bản phân phối, ví dụ: .NET Client Profile).

+ Hệ quản trị CSDL MSSQL Server 2008 hoặc mới hơn (chỉ yêu cầu khi sử dụng ở chế độ không có mạng).

- Ứng dụng Web (Web Desktop Application)

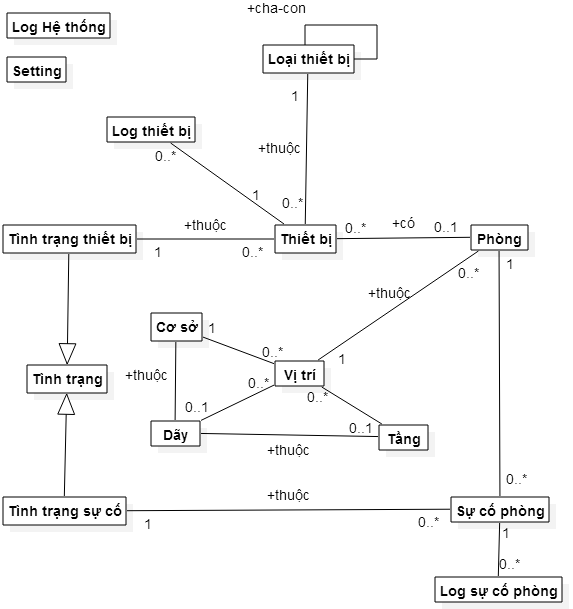
+ Chạy tốt trên trình duyệt Chrome 28, Firefox 29, Internet Explorer (IE) 8 hoặc mới hơn.

+ Hỗ trợ thiết bị di động.

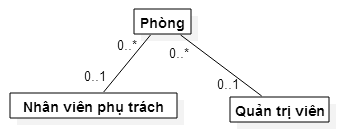
+ Vận hành tốt trên máy chủ IIS ASP.NET 4.0 hoặc mới hơn.

## 1.2. Lược đồ quan niệm

- Do sự dàn trải rộng của sơ đồ, nên sẽ trình bày từng sơ đồ con. Các sơ đồ thiết kế sau đây được trình bày theo chuẩn UML 2.0 [10].



Sơ đồ 1.1: Sơ đồ quan niệm quan hệ phòng - thiết bị



Sơ đồ 1.2: Sơ đồ quan niệm nhân viên phụ trách phòng



Sơ đồ 1.3: Sơ đồ quan niệm quan hệ phân quyền

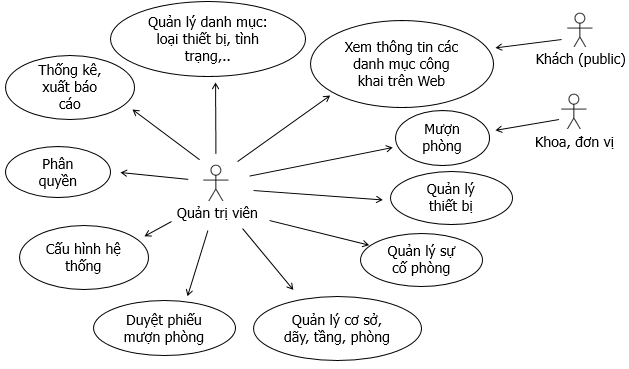


Sơ đồ 1.4: Sơ đồ quan niệm quan hệ hình ảnh



Sơ đồ 1.5: Sơ đồ quan niệm quan hệ mượn phòng

## 1.3. Lược đồ trường hợp sử dụng (use case)

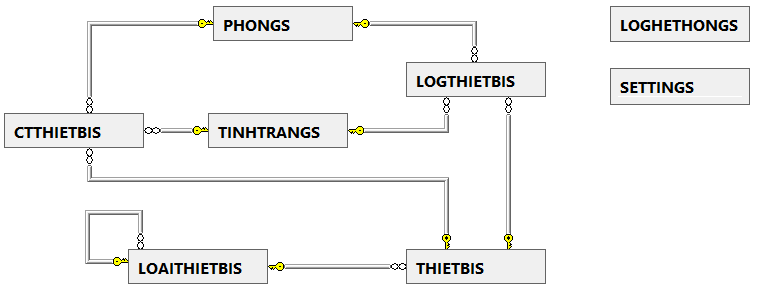


Sơ đồ 1.6: Sơ đồ trường hợp sử dụng (use case)

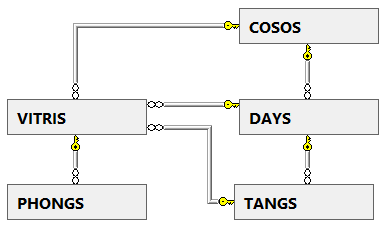
## 1.4. Lược đồ CSDL mức vật lý

\*Các lược đồ CSDL được thể hiện trên hệ quản trị MSSQL Server 2008 R2.

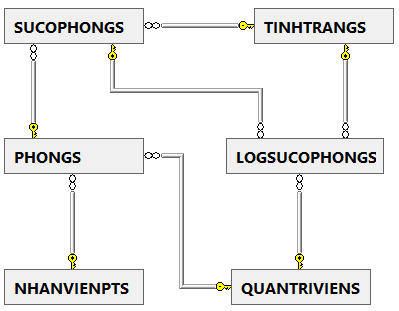
- Khi EF làm việc, bản thiết kế lớp sẽ được tự động ánh xạ thành cấu trúc CSDL mức vật lý.



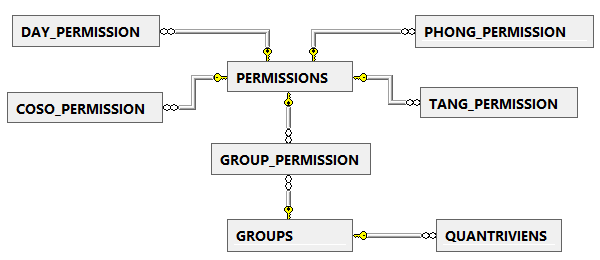
Sơ đồ 1.7: Sơ đồ CSDL vật lý quan hệ phòng - thiết bị



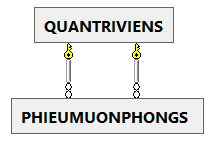
Sơ đồ 1.8: Sơ đồ CSDL vật lý quan hệ phòng - vị trí



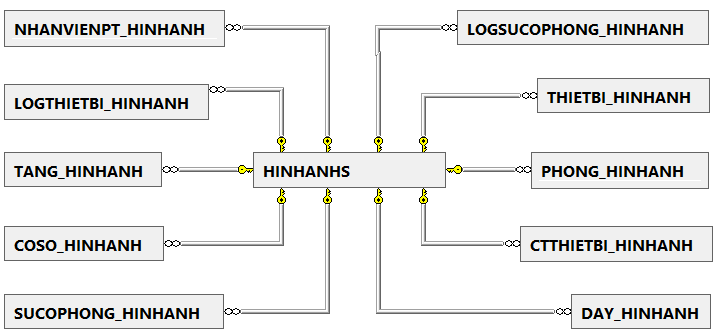
Sơ đồ 1.9: Sơ đồ CSDL mức vật lý quan hệ phòng - sự cố - nhân viên phụ trách



Sơ đồ 1.10: Sơ đồ CSDL mức vật lý quan hệ phân quyền



Sơ đồ 1.11: Sơ đồ CSDL mức vật lý quan hệ phiếu mượn phòng



Sơ đồ 1.12: Sơ đồ CSDL mức vật lý quan hệ hình ảnh

- Chi tiết các bảng trong CSDL vật lý.

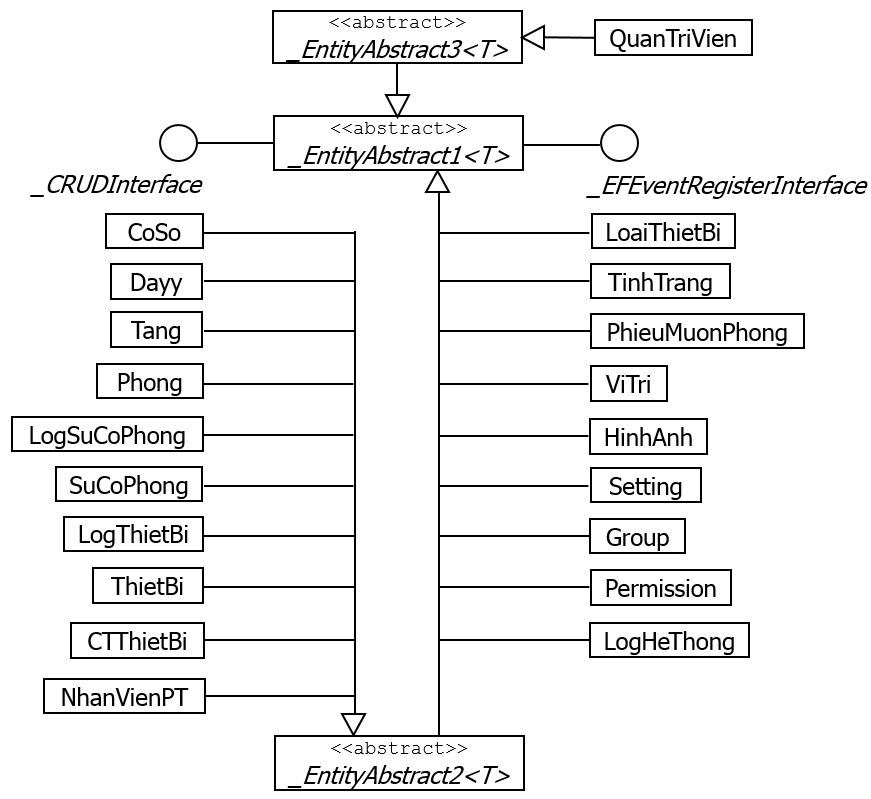
(Xem phụ lục A)

## 1.5. Lược đồ lớp (class)

### 1.5.1. Lược đồ các lớp thực thể

#### 1.5.1.1. Sơ đồ kế thừa giữa các lớp

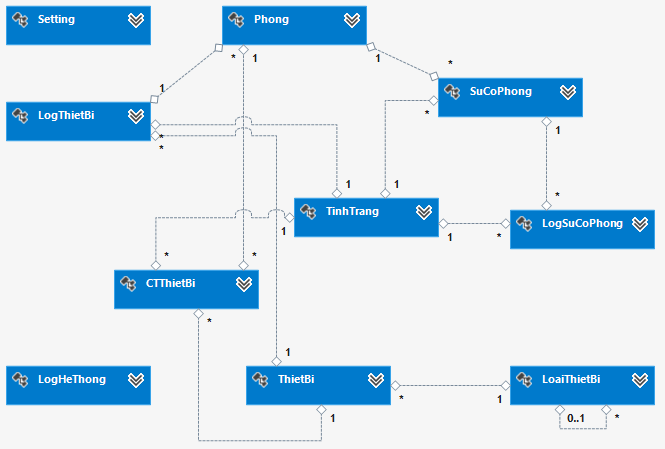
- Giữa các thực thể có nhiều thuộc tính giống nhau cần lưu trữ. Giữa các thực thể có sự giống nhau trong các chức năng nghiệp vụ, nên việc sử dụng mô hình kế thừa trong thiết kế là điều vô cùng cần thiết. Điều đó khiến cho việc bảo trì các đoạn mã được diễn ra nhanh chóng và tăng tính nhất quán trong toàn cấu trúc.



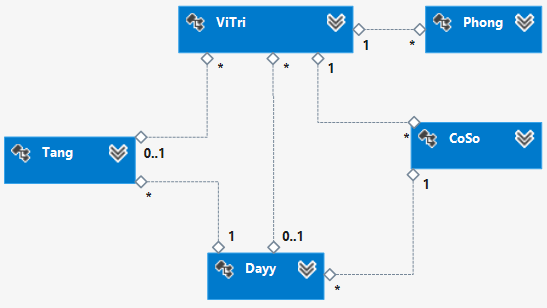
Sơ đồ 1.13: Sơ đồ kế thừa các lớp thực thể

- Một mô hình kế thừa tốt là mô hình mà trong đó các đoạn mã định nghĩa thuộc tính cũng như các đoạn mã lập trình được đặt đúng vị trí, đúng mức (level) cần thiết để tận dụng tối đa mã nguồn đã có, tránh trùng lắp.

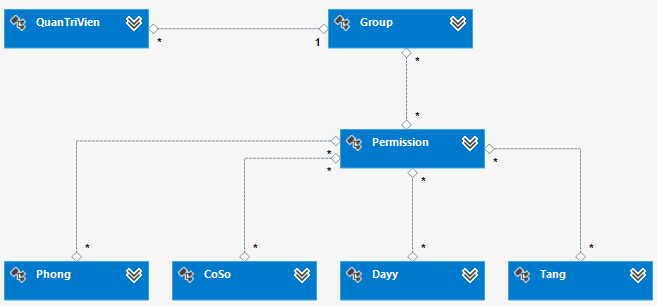
#### 1.5.1.2. Sơ đồ quan hệ giữa các lớp



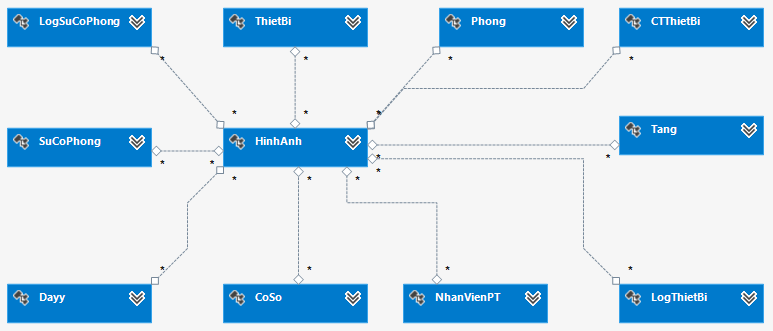
Sơ đồ 1.14: Sơ đồ lớp quan hệ phòng - thiết bị



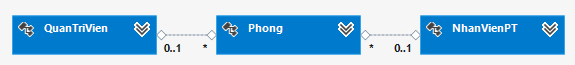
Sơ đồ 1.15: Sơ đồ lớp quan hệ phòng - vị trí



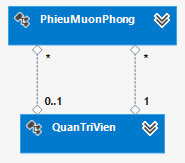
Sơ đồ 1.16: Sơ đồ lớp quan hệ phân quyền



Sơ đồ 1.17: Sơ đồ lớp quan hệ hình ảnh

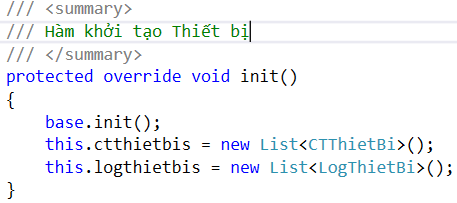


Sơ đồ 1.18: Sơ đồ lớp quan hệ phân công phòng



Sơ đồ 1.19: Sơ đồ lớp quan hệ mượn phòng

- Khi các lớp cứng (concreted class) kế thừa từ lớp ảo, chúng sẽ có được các chức năng sẵn có mà không cần phải lập trình lại, nếu vì lý dó nào đó mà các lớp cứng muốn bổ sung các đoạn mã phụ, thì sẽ gọi đè (override).



Hình 1.1: Minh họa cách override một phương thức từ lớp cứng

- Chi tiết bản thiết kế các giao diện và lớp ảo.

(Xem phụ lục B)

- Chi tiết bản thiết kế các lớp cứng.

(Xem phụ lục C)

### 1.5.2. Các lớp thư viện liên quan

- Ngoài các lớp thực thể chính thì xoay quanh còn có lớp các thư viện hỗ trợ, các thư viện trợ giúp (helper), cấu hình liên quan...

\*Vì lý do số lượng lớp thư viện quá nhiều nên sau đây chỉ liệt kê tên và mô tả của từng lớp.

- Thư viện trợ giúp CSDL (không gian tên: PTB.Global):

+ client\_database: cung cấp các phương thức giao tiếp với CSDL ở máy chủ địa phương.

+ server\_database: cung cấp các phương thức giao tiếp với CSDL ở máy chủ tập trung tập trung.

+ working\_database: cung cấp các phương thức giao tiếp với CSDL không phụ thuộc máy chủ.

- Thư viện trợ giúp (không gian tên: PTB.Libraries): DatabaseHelper, EmailHelper, ImageHelper, PermissionHelper.

- Thư viện trợ giúp (không gian tên: SHARED.Libraries): DatabaseHelper, DateTimeHelper, EmailHelper, FileHelper, FTPHelper, HTTPHelper, ImageHelper, MobileDetect, PermissionHelper, ReportHelper, ServerTimeHelper, SkinHelper, StringHelper, Debug, GUID.

- Thư viện trợ giúp truy xuất CSDL (không gian tên: PTB.Entities): OurDBContext, DBInstance.

- Thư viện trợ giúp lọc, hiển thị dữ liệu (không gian tên: PTB.DataFilter): \_FilterAbstract, ChiTietTBHienThi, DayyFilter, DoSearch, LoaiTBHienThi, NhanVienPTFilter, ...

- Thư viện cấu hình (không gian tên: PTB.Global):

+ remote\_setting: các cấu hình áp đặt cho toàn máy trạm.

* ftp\_host: các cấu hình để tải các tập tin lên máy chủ.
* http\_host: các cấu hình để tải tập tin từ máy chủ về.
* smtp\_config: các cấu hình gửi mail.
* email\_template: các cấu hình cài đặt template cho mail.

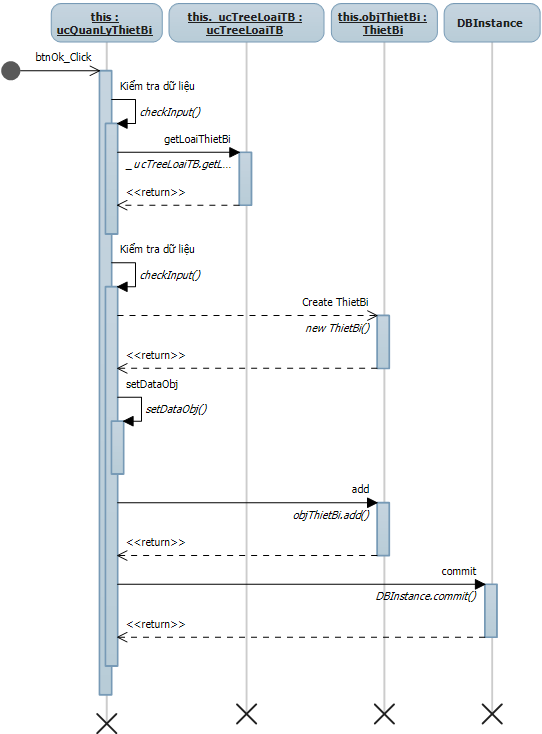
+ local\_setting: các cấu hình áp đặt lên máy trạm hiện hành.

+ sync: các cấu hình đồng bộ CSDL.

## 1.6. Lược đồ tuần tự (sequences)

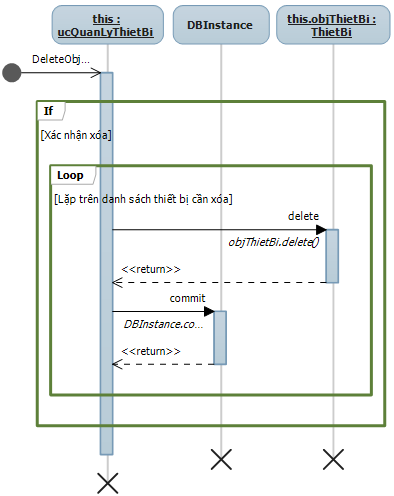
- Do hạn chế về không gian trình bày cũng như không thể đưa hết mọi bản thiết kế vào trong báo cáo này nên nhóm chọn ra 5 chức năng chính để trình bày.

### 1.6.1. Thêm thiết bị mới



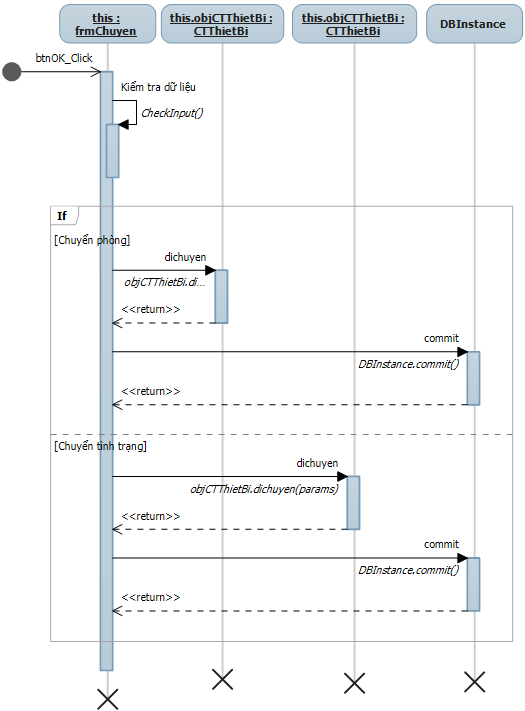
Sơ đồ 1.20: Sơ đồ tuần tự cho chức năng "Thêm thiết bị mới"

### 1.6.2. Xóa thiết bị khỏi hệ thống



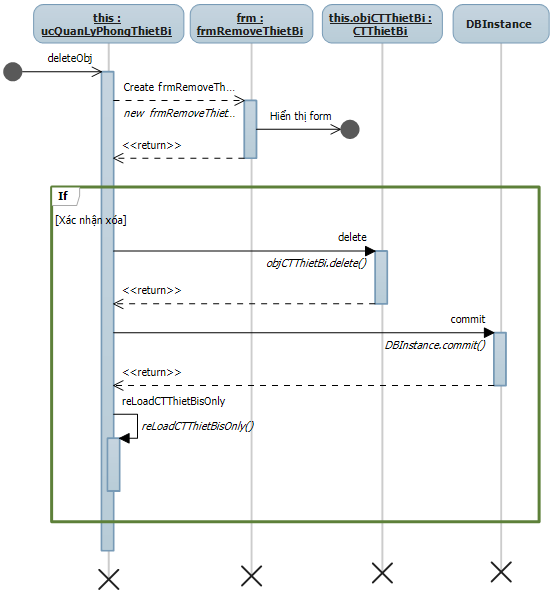
Sơ đồ 1.21: Sơ đồ tuần tự cho chức năng "Xóa thiết bị khỏi hệ thống"

### 1.6.3. Chuyển thiết bị giữa các phòng hoặc chuyển tình trạng thiết bị



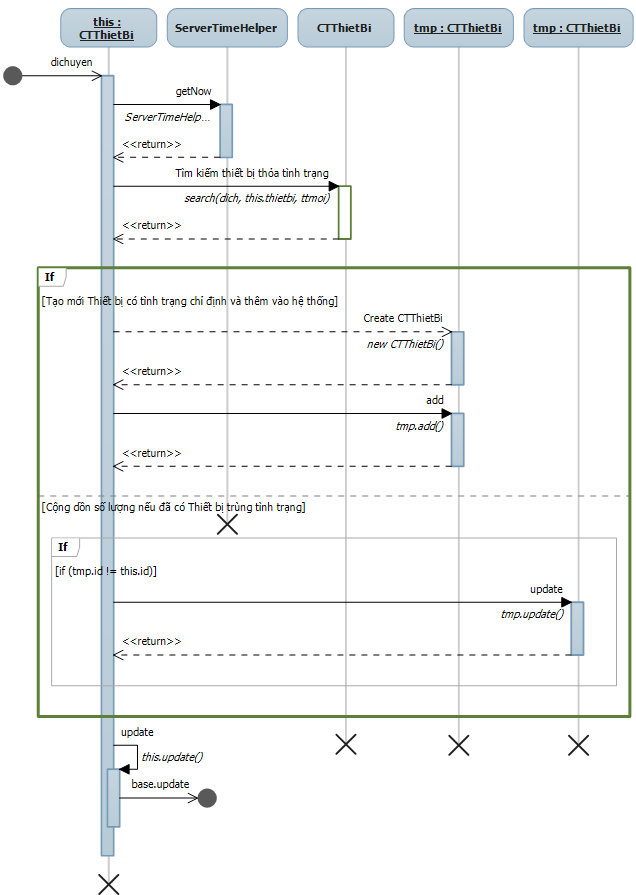
Sơ đồ 1.22: Sơ đồ tuần tự cho chức năng "Chuyển thiết bị giữa các phòng hoặc chuyển tình trạng thiết bị"

### 1.6.4. Loại bỏ thiết bị khỏi phòng



Sơ đồ 1.23: Sơ đồ tuần tự cho chức năng "Loại bỏ thiết bị khỏi phòng"

### 1.6.5. Chức năng chuyển tình trạng thiết bị



Sơ đồ 1.24: Sơ đồ tuần tự cho chức năng "Chuyển tình trạng thiết bị"

CHƯƠNG 2: THỰC THI

## 2.1. Môi trường lập trình và phát triển ứng dụng

- Hệ điều hành: Windows 7.

- Ngôn ngữ lập trình: C#.

- Nền tảng .NET Framework 4.0.

- Công cụ hỗ trợ soạn thảo và biên dịch: Visual Studio 2012.

- Hệ quản trị CSDL MSSQL Server Express 2008.

- Máy chủ web IIS 8.0 Express.

- Thư viện MS Sync Framework 2.1.

- Thư viện Enitity Framework 6.1.1

- Thư viện DevExpress 13.2.9.

- Trình quản lý mã nguồn (source control): Git từ nhà cung cấp Github, Inc.

## 2.2. Mô hình tổ chức ứng dụng

### 2.2.1. Tổ chức ứng dụng theo hướng module hóa trong lập trình đa nền tảng (cross - platform)

- Công nghệ .NET có thể được vận hành trên nhiều nền tàng hệ thống, trong đó phổ biến nhất là trên nền Windows Desktop Application, Windows Mobile Application, Windows Web Server IIS và thậm chí trên một số hệ thống hiếm như .NET MonoDevelop trên MAC OS,...

- Do đó, việc tổ chức các thành phần (module) riêng biệt sẽ giúp tận dụng được lại mã nguồn cho các ứng dụng chạy trên các nền tảng khác nhau. Đồng thời tăng tính thống nhất trong chương trình.

- Mô hình phân tích hướng module:

+ Mô hình quan niệm:



Sơ đồ 2.1: Mô hình quan niệm tổ chức ứng dụng hướng module hóa

+ Mô hình chi tiết:



Sơ đồ 2.2: Mô hình chi tiết tổ chức ứng dụng hướng module hóa

### 2.2.2. Mô hình 3 lớp trong ứng dụng hướng dữ liệu

- Các ứng dụng mà công việc diễn ra ở nhiều tầng khác nhau, trong đó chức năng của mỗi tầng hầu như toàn độc lập thì việc tách các tầng này ra thành các lớp chức năng sẽ khiến cho công việc bảo trì và nâng cấp diễn ra nhanh chóng và gọn nhẹ. Ứng dụng hướng dữ liệu là một trong những dạng này [11, tr.66-71].

- Mô hình 3 lớp phổ biến bao gồm:

+ Tầng truy xuất dữ liệu (Data Access Layer - DAL).

+ Tầng nghiệp vụ chức năng (Bussiness Logic Layer - BLL).

+ Tầng hiển thị dữ liệu (Presentation Layer - PL).

- Tùy thuộc vào điều kiện sẵn có của hệ thống mà ta sẽ sử dụng những tầng đã có và chỉ thiết kế cho những tầng chưa có.

- Cụ thể trong ứng dụng này, khi áp dụng công nghệ EF thì mặc định đã có sẵn tầng truy xuất dữ liệu DAL, nên ta chỉ việc kết hợp với 2 tầng còn lại của ứng dụng là được.

- 2 tầng trong ứng dụng này bao gồm:

+ Tầng nghiệp vụ chức năng (Bussiness Logic Layer - BLL): chính là các lớp thực thể và các nghiệp vụ được định nghĩa trên thực thể. Đây là trái tim của hệ thống, mọi thao tác logic về xử lý và điều khiển luồng dữ liệu đều được diễn ra ở đây.

+ Tầng hiển thị dữ liệu (Presentation Layer - PL): chính là các giao diện chức năng ứng với các nền tảng Windows Desktop hay Web,...Hiện ứng dụng có 3 lớp hiển thị ở tầng này (lớp ứng dụng Windows Desktop, lớp ứng dụng Web Desktop và lớp ứng dụng Web Mobile dành cho thiết bị di động).

### 2.2.3. Mô hình MVP (Model - View - Presenter) dành cho ứng dụng WinForm Desktop

- Trong hầu hết trường hợp, các giao diện cần được tái sử dụng nhằm đảm bảo tính thống nhất và bảo trì mã nguồn thuận lợi. Khi đó cần tìm giải pháp tách riêng giao diện ra khỏi các thành phần xử lý. Mô hình MVP là một trong những giải pháp đó.

- Các tính năng của MVP:

+ Tận dụng lại các thiết kế giao diện.

+ Tận dụng lại các logic lập trình trên giao diện.

+ Tạo sự thống nhất trong giao diện giữa các dự án (project).

- Mô hình MVP:



Sơ đồ 2.3: Mô hình MVP

- Trình hiển thị tùy biến (Custom User Control): coi như là một module MVP nhỏ được tái sử dụng cho nhiều khu vực. Các Custom User Control này không có sẵn trong các bộ công cụ đi kèm mà bản thân người lập trình phải tạo riêng nhằm phục vụ các mục đích chuyên biệt.

### 2.2.4. Mô hình ASP.NET Web Forms dành cho ứng dụng Web

- Hiện DevExpress hỗ trợ ASP.NET Web Forms tốt hơn ASP.NET MVC, do đó để sử dụng được các công nghệ của DevExpress dành cho nền tảng Web thì ASP.NET Web Forms là lựa chọn phù hợp nhất.

- Các tính năng khác của ASP.NET Web Forms:

+ Hỗ trợ Custom User Control.

+ Do đặc điểm Code behind (lập trình tách biệt khỏi giao diện) theo kiểu Event- driven (hướng sự kiện) sẽ thích hợp cho những tác vụ đòi hỏi giao tiếp nhiều giữa máy chủ và máy khách.

## 2.3. Các công nghệ và kỹ thuật lập trình được áp dụng

### 2.3.1. Công nghệ Entity Framework (EF) trong lập trình dữ liệu hướng đối tượng (OOP)

#### 2.3.1.1. Tổng quan

- EF Là một DB ORM (Database Object - Relational Mapping) giúp ánh xạ cơ sở dữ liệu quan hệ lên một khung nhìn các đối tượng và tập hợp các đối tượng [5].

- EF là một mã nguồn mở được chính Microsoft phát triển dựa trên nền tảng .NET.

- EF giúp người lập trình tiết kiệm được rất nhiều thời gian và công sức trong việc truy xuất và làm việc với các CSDL quan hệ, bởi vì bản thân EF đã cung cấp đầy đủ các tính năng và công nghệ tiên tiến giúp độc lập hóa khối CSDL ra khỏi lớp truy xuất.

- Được ứng dụng nhiều trong các nền tảng dạng Domain - Driven Design, cho phép người lập trình ảo hóa CSDL vật lý, giảm sự phục thuộc vào CSDL vật lý, từ đó lập trình viên có thể làm việc trong suốt với CSDL nền, không quan tâm Hệ quản trị CSDL đích hay phiên bản khác nhau, miễn được EF hỗ trợ.

- Phiên bản mới nhất: 6.1.1.

- Phiên bản sử dụng trong đề tài: 6.1.1.

#### 2.3.1.2. Mô hình triển khai Code first

- Thiết kế cấu trúc CSDL bằng cách định nghĩa các lớp (class) trước (Code first): là một cách tiếp cận mới trong việc thiết kế cho các ứng dụng hướng CSDL, thay vì thiết kế cấu trúc CSDL vật lý trước thì Code first cung cấp các đặc tả để người lập trình có thể định nghĩa cấu trúc CSDL bằng các lớp và các logic, ràng buộc trên lớp; tận dụng được các tính năng kế thừa trên lớp. Sau đó EF sẽ tự động ánh xạ bản thiết kế xuống cấu trúc CSDL.

- EF hỗ trợ 2 giải pháp làm việc với CSDL.

+ Giải pháp 1 (Code first to existing Database): đối với CSDL đã có từ trước, người lập trình nếu muốn áp dụng Code first thì cần phải tìm cách đưa bản thiết kế CSDL vật lý lên mức lớp trong mã nguồn, sau đó chỉ định bản thiết kế lớp này ứng với cấu trúc CSDL vật lý hiện tại. Kể từ thời điểm này, thiết kế sẽ được thực hiện trên mức lớp trước, sau đó sẽ được ánh xạ lại trong cấu trúc CSDL vật lý.

+ Giải pháp 2 (Code first to new Database): nếu chưa có bản thiết kế CSDL vật lý sẵn, thì lập trình viên chỉ cần tạo bản thiết kế lớp trong mã nguồn như mong muốn, sau đó gọi trình ánh xạ của EF để tạo cấu trúc CSDL vật lý.

\*Đề tài này do được nghiên cứu mới hoàn toàn nên giải pháp 2 được áp dụng.

#### 2.3.1.3. Ánh xạ TPC (Table Per Concrete class) trong kế thừa thuộc tính

- Định nghĩa: TPC là cách thiết kế mà trong đó mỗi lớp cứng sẽ ánh xạ thành một bảng trong CSDL vật lý. Tận dụng được các đoạn mã logic bằng cách thiết kế mô hình kế thừa, đa hình [12].

- Trong TPC, các lớp ảo (abstract) sẽ không được ánh xạ xuống CSDL vật lý, các lớp này chỉ dùng cho mục đích kế thừa. Muốn ánh xạ một thực thể xuống CSDL vật lý thì lớp thực thể đó bắt buộc phải là một lớp cứng.



Sơ đồ 2.4: Mô hình triển khai TPC trong kế thừa

#### 2.3.1.4. Các cách biểu diễn quan hệ 1 - n hoặc n - n trong EF

- Gọi tập hợp A = {Entity A0, Entity A1,...Entity Am} chứa các thực thể ở quan hệ n.

- Gọi tập hợp B = { Entity B0, Entity B1,...Entity Bn} chứa các thực thể ở quan hệ 1.

- Một tích Đề-các (Descartes) diễn tả mối quan hệ 1 - n hoặc n - n từ Bi đến Aj (i < [B], j < [A]):

B x A { (b,a) | b ∈ B, a ∈ A }

- Mối quan hệ 1 - n trong CSDL quan hệ có thể được biểu diễn bằng 2 cách sau:

+ Cách 1: nâng cấp quan hệ 1 - n thành quan hệ n - n và ngầm định không sử dụng chiều tham chiếu ngược lại: khi đó mỗi quan hệ (B x A)i ( i < [B x A] ) sẽ liên kết với (1+1)+1=3 bảng vật lý.

=> Tổng các bảng vật lý tối thiểu cần thiết để biểu diễn quan hệ trên (B x A) là:

Sum = [A] + [B] + [B x A]

=> Phù hợp với trường hợp mối quan hệ cần có thêm các thuộc tính kèm theo.

+ Cách 2: truyền thống, đối tượng ở quan hệ n sẽ có n khóa ngoại trỏ đến đối tượng ở quan hệ 1: khi đó mỗi quan hệ (B x A)i (i < [B x A]) sẽ liên kết ứng với 1+1=2 bảng vật lý.

=> Tổng các bảng vật lý tối thiểu cần thiết để biểu diễn quan hệ trên (B x A) là:

Sum = [A] + [B]

\*EF làm việc được với cả 2 cách biểu diễn trên. Tùy thuộc vào từng trường hợp cụ thể mà lựa chọn phương án thích hợp.

- Mối quan hệ n - n trong CSDL quan hệ có thể được biểu diễn bằng 2 cách:

+ Cách 1: tương tự cách 1 trong biểu diễn quan hệ 1 - n (nhưng vì đã là quan hệ n- n nên không cần nâng cấp).

+ Cách 2: sử dụng thuộc tính đa trị trong từng trường định nghĩa khóa ngoại, tuy nhiên sẽ vi phạm dạng chuẩn CSDL 1NF (dạng chuẩn thấp nhất) do chứa thuộc tính đa trị, và không được EF hỗ trợ, nên không khả thi khi triển khai.

\*Nhóm sử dụng cách 1 trong việc biểu diễn quan hệ n - n giữa: Cơ sở, Dãy, Tầng, Phòng,... và Hình Ảnh.

#### 2.3.1.5. Công nghệ truy vấn LINQ

- LINQ to Entities: EF sử dụng LINQ để truy vấn trên CSDL. LINQ là một công nghệ truy vấn CSDL không phụ thuộc vào dữ liệu nguồn, tức là có thể dùng LINQ để truy vấn và thực thi các thao tác dữ liệu trên: tập hợp/danh sách các đối tượng, tập tin XML hay các Hệ quản trị CSDL khác nhau.

- LINQ IEnumberable và biểu thức chính quy Lambda:

+ LINQ IEnumberable: là một giao diện (interface) định nghĩa trong LINQ, cho phép tạo ra các câu truy vấn lồng nhau trên một tập hợp có thực thi lớp giao diện này. Kết quả của một loạt các thực thi sẽ được trả về ngay sau khi có lời gọi để chuyển từ tập hợp không chính quy (non - generic) sang tập hợp chính quy (generic). Đây là tính năng rất hay trong LINQ, mà nhờ đó tiết kiệm được chi phí cũng như thời gian thực thi dữ liệu, vì câu truy vấn cuối cùng chỉ được kết lại và gọi chạy khi hoàn tất một loạt các truy vấn lồng nhau thay vì phải tải dữ liệu lên sau mỗi bước nạp điều kiện.

+ Lambda là một dạng biểu thức chính quy được sử dụng trong các truy vấn đến các lớp dữ liệu có đăng ký thực thi lớp IEnumberable.



Sơ đồ 2.5: Cách hoạt động của kỹ thuật truy vấn lồng trong LINQ

#### 2.3.1.6. Các tính năng khác của EF

- Trì hoãn tải dữ liệu (Lazy loading): trì hoãn tải dữ liệu khi chưa cần thiết, dữ liệu chỉ được tải khi có lời gọi tới lần đầu tiên (giảm thời gian nạp dữ liệu ban đầu). Lazy loading rất hữu ích trong trường hợp một các đối tượng có mối quan hệ phức tạp qua lại với nhau và cây liên hệ có chiều cao lớn hơn 1.

- Mặc định EF được cấu hình bật Lazy loading.



Sơ đồ 2.6: Tham chiếu ngược trên các quan hệ 1 - n, n - n

- Phiên bản CSDL (Database Version - Upgrade/Downgrade/Rebase): EF xem mỗi sự thay đổi trong bản thiết kế lớp (class) sẽ tương ứng với một phiên bản CSDL mức vật lý. Khi bản thiết kế có sự thay đổi, EF sẽ ghi nhận lại sự thay đổi đó và tạo ra các đoạn mã để điều chỉnh cấu trúc CSDL hiện tại lại cho khớp với bản thiết kế mới. Những đoạn mã này được gọi là một phiên bản của CSDL, và sẽ được định danh bằng tên mã nhằm phục vụ cho quá trình dịch chuyển phiên bản.

- Dịch chuyển phiên bản CSDL (Migration to Database Version):

+ Nâng cấp (Upgrade): là hình thức dịch chuyển từ phiên bản thấp Vi lên phiên bản cao hơn Vj (i<j).

+ Hạ cấp (Downgrade): là hình thức di chuyển từ phiên bản cao Vj xuống phiên bản thấp hơn Vi (i<j).

\*EF cho phép nhảy cốc giữa các phiên bản. Ví dụ: có thể nhảy từ phiên bản v3 xuống v1 mà không cần thông qua v2 và ngược lại, miễn là các v1, v2, v3 đã được định danh trước đó. Tính năng này rất hữu ích khi dự án được thực hiện ở quy mô nhóm, khi đó các lập trình viên sẽ tiết kiệm được rất nhiều thời gian cho khâu cập nhật CSDL trong suốt quá trình lập trình và kiểm thử.

- Sự tương thích giữa các phiên bản CSDL: đây cũng là vẫn đề được quan tâm tới trong khi thiết kế và vận hành ứng dụng. EF chỉ có thể làm việc trên phiên bản CSDL Vj nếu Vj tương thích với phiên bản CSDL hiện tại (ứng với thiết kế lớp (class) hiện tại). Tùy thuộc vào sự thay đổi giữa các phiên bản mà có thể tương thích hoặc không tương thích cụ thể như sau:

- Xét 2 phiên bản CSDL Vi và Vj (i<j):

+ Tương thích ngược: Vj được xem là tương thích ngược với Vi nếu CSDL Vj bao trùm CSDL Vi.



Sơ đồ 2.7: Tương thích ngược phiên bản CSDL

+ Tương thích xuôi: Vi được xem là tương thích xuôi với Vj nếu CSDL Vi bao trùm CSDL Vj (điều này rất hiếm khi xảy ra vì đa phần bản thiết kế mới Vj luôn mở rộng hơn so với bản cũ Vi).



Sơ đồ 2.8: Tương thích xuôi phiên bản CSDL

+ Không tương thích: trong các trường hợp còn lại.



Sơ đồ 2.9: Không tương thích phiên bản CSDL (trường hợp 1)



Sơ đồ 2.10: Không tương thích phiên bản CSDL (trường hợp 2)

\*Các ví dụ minh họa trên chỉ thể hiện ở mức đơn giản nhất, trên thực tế việc xét tính tương thích sẽ phức tạp hơn nhiều do sự kết hợp của nhiều bảng và định nghĩa các thuộc tính trên bảng.

- Kiểm định mô hình (Model checking): kiểm tra cấu trúc CSDL có bị thay đổi bởi các tác nhân ngoài hệ thống hay không, đảm bảo hệ thống làm việc ổn định và đúng đắn. Model cheking là cực kỳ quan trọng khi làm việc với EF bởi vì các trường thuộc tính của dữ liệu vật lý gắn chặt với các lớp tương ứng của ứng dụng khi bộ máy ánh xạ hoạt động, nên chỉ với một thay đổi nhỏ về định nghĩa của CSDL vật lý cũng sẽ khiến EF không hoạt động.

- Trình khởi tạo CSDL tùy biến (Custom Database Initalizer) [3]: chỉ định phương thức khởi tạo CSDL, EF cho phép lựa chọn và tùy biến nhiều chỗ trong quá trình tạo tự động cấu trúc CSDL khi ánh xạ bản thiết kế xuống CSDL mức vật lý, các tính năng bao gồm: tự động tạo CSDL nếu chưa có ?, tự động tạo bảng nếu chưa có ?, tự động tạo dữ liệu mẫu ban đầu ?,…

+ CreateDatabaseIfNotExists: mặc định của EF. Tự động tạo CSDL nếu chưa có.

+ DropCreateDatabaseIfModelChanges: tự động tạo lại CSDL khi cấu trúc bị thay đổi. Tuy nhiên không an toàn vì chỉ một sơ suất nhỏ trong khâu thiết kế cũng sẽ dẫn đến nguy cơ mất CSDL.

+ DropCreateDatabaseAlways: luôn luôn xóa và tạo mới lại CSDL mỗi khi ứng dụng khởi chạy, phù hợp với các ứng dụng sử dụng CSDL như là một bộ nhớ tạm trong lúc làm việc, và muốn CSDL rỗng cho mỗi phiên mới.

+ Custom DB Initializer: nếu các định nghĩa trên không đáp ứng được yêu cầu thì người lập trình có thể tự định nghĩa một trình khởi tạo dữ liệu riêng cho CSDL.

- Tạo dữ liệu mẫu mặc định (Data Seeding): cung cấp các đặc tả để tạo dữ liệu mặc định ban đầu khi tạo mới một CSDL, ví dụ: tài khoản quản trị mặc định, các giá trị cài đặt mặc định,...

- Mô hình dữ liệu hướng sự kiện (Event-Driven model - BootStrapper): tương tự như mô hình lập trình giao diện hướng sự kiện được áp dụng trong Winform, mô hình dữ liệu hướng sự kiện cũng cho phép các đối tượng tham gia đăng ký các sự kiện xảy ra khi hệ thống theo dõi của EF làm việc. Được ứng dụng trong các nghiệp vụ như: sau khi cập nhật thành công thì trường "date\_modified" sẽ có giá trị thời gian hiện tại, sau khi sửa đổi một đối tượng sẽ ghi nhật ký hệ thống...Các nghiệp vụ này sẽ được thực thi một cách tuần tự và chính xác như mong muốn của lập trình viên.



Sơ đồ 2.11: Giao tiếp 2 chiều trong mô hình dữ liệu hướng sự kiện

- Khai báo proxy cho các thuộc tính (property virtual proxy): EF đòi hỏi thuộc tính khóa ngoại của các lớp phải được khai báo dạng ảo (virtual) để EF được phép tạo proxy ẩn trong các kỹ thuật theo dõi hay Lazy loading khi các thuộc tính này được truy xuất hoặc sửa đổi.

- Trình quản lý giao dịch (Transaction Manager) trong các kỹ thuật quay ngược (rollback): trong một ứng dụng lớn thì số tác vụ con thực thi trong một nghiệp vụ cụ thể là nhiều, hệ quản trị CSDL chỉ xử lý đơn nguyên dữ liệu (hoặc là tất cả thao tác trên dữ liệu đều được thực thi hoặc là không) ở mức rất thấp (mức tác vụ INSERT, DELETE,... trên từng đối tượng). Do đó, nếu muốn đảm bảo tính đơn nguyên ở mức cao hơn (cả một nghiệp vụ hoàn chỉnh) thì việc điều khiển và gọi trình quản lý giao dịch đúng thời điểm sẽ giải quyết được bài toán "đảm bảo toàn vẹn CSDL". EF cung cấp một cơ chế Transaction rất đơn giản và hiệu quả.

- Trạng thái của đối tượng (Entity State) và tính năng cập nhật chọn lọc: để có thể theo dõi được các sự thay đổi dữ liệu trên các đối tượng trong lúc thực thi (runtime), EF đưa ra định nghĩa về trạng thái của các đối tượng, trong đó một đối tượng có thể thuộc một trong các trạng thái sau:

+ Attached: đối tượng mới khởi tạo và được đưa và hệ thống theo dõi, tuy nhiên chưa được lưu xuống CSDL.

+ Detached: đối tượng đã bị loại khi hệ thống theo dõi.

+ Added: đối tượng được đưa vào hàng đợi, chờ thêm vào CSDL.

+ Modified: đối tượng được đánh dấu là đã bị thay đổi ít nhất 1 thuộc tính, được đưa vào hàng đợi chờ cập nhật xuống CSDL.

+ Unchanged: đối tượng được đánh dấu là sạch, có thể là mới được khởi tạo hoặc là mới được tải lên từ CSDL.

+ Deleted: đối tượng được đánh dấu là bị xóa, được đưa vào hàng đợi chờ xóa khỏi CSDL.

- Trình lọc dữ liệu (DataFilter) trong hiển thị dữ liệu: một đối tượng chỉ mang các thuộc tính trực tiếp của bản thân nó, trong lập trình giao diện, thông thường khi hiển thị thông tin một đối tượng nào đó, ta thường hiển thị các thuộc tính gián tiếp (thuộc tính của khóa ngoại). Do đó trình xử lý giao diện sẽ không làm việc trực tiếp với các đối tượng này mà làm việc thông qua một lớp mặt nạ (mask) gọi là lớp lọc dữ liệu (DataFilter), nhiệm vụ của DataFilter là kết các đối tượng có liên quan lại với nhau sau đó chọn ra các thuộc tính cần hiển thị.

- Ngữ cảnh CSDL (Database Context) và cơ chế hoạt động lớp truy xuất CSDL ( Singleton Database Instance Provider):

+ EF xem Database Context là một ngữ cảnh truy xuất đến CSDL, trên đó chứa các định nghĩa về nguồn dữ liệu. Database Context là không gian làm việc của EF. Có thể có nhiều Database Context được định nghĩa trên cùng một CSDL.

+ Singleton giúp các lớp thực thể nhìn thấy cùng một Database Context trong suốt phiên làm việc, bởi vì EF đòi hỏi các đối tượng sinh ra từ các lớp thực thể phải thống nhất về Database Context, một đối tượng không thể được theo dõi bởi các Database Context khác nhau.

+ Phiên làm việc được đánh dấu từ lúc Database Context được khởi tạo cho đến khi bị hủy bỏ (dispose), các đối tượng nằm ngoài phiên làm việc được xem là không hợp lệ và không có ý nghĩa về mặt dữ liệu, muốn làm việc lại trên các đối tượng này nhất thiết phải được tải lại trong một phiên làm việc Database Context khác.



Sơ đồ 2.12: Cách hoạt động giữa Singleton và Database Context

### 2.3.2. Công nghệ Sync Framework của Microsoft trong đồng bộ CSDL tập trung

#### 2.3.2.1. Tổng quan

- Định nghĩa: Sync Framework là công nghệ được Microsoft phát triển với mục đích chính là đồng bộ dữ liệu qua lại giữa các nguồn dữ liệu, dữ liệu nguồn ở đây có thể là hệ thống tập tin hoặc là một CSDL của một hệ quản trị nào đó, hiện Sync Framework hỗ trợ các dữ liệu nguồn tương thích chuẩn ADO.NET mà trong đó Hệ quản trị MSSQL Server hoàn toàn đáp ứng được các yêu cầu trên [6].

- Đồng bộ dữ liệu giữa các CSDL ở mức đơn vị dữ liệu là bảng.

- Sử dụng kỹ thuật trigger trên từng bảng, trigger có nhiệm vụ thu thập và bắt các sự thay đổi về dữ liệu trên CSDL, sau đó lưu trữ lại trong các bảng theo dõi (tracking table, mỗi bảng được chỉ định trong Sync Scope sẽ phát sinh ra một bảng theo dõi tương ứng) mà Sync Framework tạo ra khi một Sync Scope được cài đặt.



Hình 2.1: Cách tổ chức lưu trữ của Sync Framework

- Nhờ các kỹ thuật lưu trữ và theo dõi dữ liệu như đã được giới thiệu ở trên mà Sync Framework sẽ chỉ đồng bộ những dữ liệu sai khác giữa các nguồn, do đó quá trình xử lý và truyền nhận dữ liệu sẽ tiết kiệm được thời gian và tài nguyên hệ thống, khác với các giải pháp truyền thống là phải tải mới toàn bộ dữ liệu.

- Cách hoạt động



Sơ đồ 2.13: Cách hoạt động của Sync Framework

- Phiên bản mới nhất hiện tại: 2.1

- Phiên bản sử dụng trong đề tài: 2.1

\*Trong phạm vi ứng dụng của đề tài này, sẽ chỉ xem xét đến nguồn dữ liệu là Hệ quản trị CSDL, cụ thể là MSSQL Server.

#### 2.3.2.2. Đồ thị tiến trình trong kỹ thuật tránh deadlock khi đồng bộ

- Đồ thị tiến trình đồng bộ: như đã đề cập ở phần trên, Sync Framework đồng bộ ở mức đơn vị dữ liệu là bảng, nên thứ tự các bảng trong một tiến trình đồng bộ là vô cùng quan trọng vì đặc thù ràng buộc khóa ngoại của CSDL quan hệ. Nếu bảng B có chứa khóa ngoại tham chiếu đến bảng A thì bảng A phải được xử lý trước bảng B.

=> Cần phải xây dựng đồ thị tiến trình cho phiên đồng bộ.

- Đặc điểm của đồ thị tiến trình đồng bộ:

+ Do thứ tự bảng được xem xét nên đồ thị là đồ thị có hướng.

+ Trong CSDL quan hệ không được phép chứa có liên hệ vòng nên đồ thị là đơn đồ thị, không chứa chu trình (vì chu trình sẽ gây chết tiến trình đồng bộ - deadlock).

+ Đồ thị có thể liên thông hoặc không liên thông tùy thuộc vào sự giao nhau giữa các tập quan hệ bảng.

+ Đồ thị có thể có nhiều đồ thị con (các thành phần liên thông).

+ Các thành phần liên thông có thể là các cây, trường hợp này cây có thể được tách ra thành nhiều cây con để xử lý song song bằng các tiến trình song song hoặc cũng có thể được xử lý tuần tự bằng một tiến trình duy nhất, tùy thuộc thiết kế đơn luồng hay đa luồng lúc thực thi.

- Xem các bảng là các đỉnh (V - vertexes).

- Xem các liên hệ khóa ngoại từ bảng này đến bảng kia là các cung (E - edges), trong đó một cung có nút con là bảng chứa khóa ngoại và nút cha là bảng tham chiếu đến.

- Ta được đồ thị có hướng G = {V, E}, trong đó:

+ Tập hợp các nút: V = {Table 0, Table 1,...Table n}.

+ Tập hợp các cung: E = {e 0 = (V i, V j), e 1 = (V p, V q),...e m = (V t, V v)} (i, j, p, q, t, v < n).

- Nguyên tắc xây dựng tiến trình đồng bộ đơn nhất:

+ Chỉ có một tiến trình P duy nhất xử lý cho toàn đồ thị.

+ Các nút trong từng đồ thị sẽ lần lượt được đưa vào danh sách hàng đợi Q sao cho mệnh đề sau luôn đúng:

"Với mọi Qi, Qj thuộc Q nếu cung e=(Qi, Qj) thuộc E thì j phải nhỏ hơn i"

(tức là nếu bảng A có chứa khóa ngoại đến bảng B thì bảng B phải được xử lý đồng bộ trước bảng A)

=> Phương pháp này có thể được thực hiện bằng giải thuật sắp xếp trong đó điều kiện so sánh là "xét cung tạo thành có thuộc đồ thị hay không".

\*Bên cạnh cách xây dựng tiến trình đơn nhất thì còn một giải pháp khác là tiến trình song song. Trong tiến trình song song, người ta quan tâm tới dạng đặc biệt của đồ thị là cây (đồ thị vô hướng nền không chứa chu trình), khi đó mỗi nhánh đồng cấp sẽ được chia thành các tiến trình con để xử lý đa luồng. Do phương pháp này phức tạp và chỉ phù hợp với ứng dụng quy mô lớn, nên nhóm chỉ dừng lại ở mức tham khảo.

- Không gian đồng bộ (Sync Scope) [7]:

+ Định nghĩa: Sync Scope được hiểu như là định nghĩa về một phiên đồng bộ trên một CSDL cụ thể, chứa các thông tin về tập hợp bảng cần đồng bộ. Một CSDL có thể có nhiều hơn một Sync Scope. Một CSDL có thể có nhiều Sync Scope được thiết lập sẵn thông qua việc mở rộng vùng lưu trữ trên CSDL bằng các bảng tạm và các procedure chức năng.

+ Cài đặt một Sync Scope lên CSDL có sẵn: là một loạt các thao tác cần thiết để khởi tạo và định danh một Sync Scope lên trên CSDL đã có sẵn dữ liệu hoặc CSDL mới hoàn toàn, trong đó việc chỉ định danh sách các bảng cần đồng bộ được xem là quan trọng nhất, khi các phương thức khởi tạo Sync Scope được gọi, Sync Framework sẽ thực hiện nhiệm vụ còn lại.

+ Gõ bỏ một Sync Scope có sẵn ra khỏi CSDL: thao tác gỡ bỏ Sync Scope sẽ ngược lại với các bước khi cài đặt trước đó. Chỉ cần chỉ định dữ liệu nguồn và tên Sync Scope, sau đó gọi phương thức gỡ bỏ, Sync Framework sẽ thực hiện nhiệm vụ còn lại. Việc gõ bỏ Sync Scope sẽ đồng nghĩa với việc CSDL sẽ không thể cung cấp phiên đồng bộ cho các trình quản lý đồng bộ, và do đó sẽ không tham gia vào hệ thống.

+ Áp đặt Sync Scope giữa các CSDL (Fetching scope among databases): là sao chép các thông tin về các định nghĩa phiên, các cấu hình liên quan,... từ một Sync Scope trên CSDL A có sẵn sang một Sync Scope mới trên CSDL mới B, khai báo rằng CSDL A và B có thể "bắt tay" được với nhau, lúc này trình quản lý đồng bộ mới có thể nhìn thấy và làm việc được trên cả 2 CSDL này, cũng có thể nói đây là bước thiết lập một cầu nối (pipeline) dữ liệu. Việc áp đặt Sync Scope từ một CSDL này lên một CSDL là rất quan trọng. Trong một tiến trình đồng bộ cụ thể thì thao tác này được thiết lập sớm nhất ngay khi có thể. Một CSDL có thể bắt tay với nhiều CSDL khác, đây là tính năng sẽ được ứng dụng trong các mô hình triển khai máy trạm khi đưa vào vận hành.

+ Hướng đồng bộ (chỉ lên/chỉ xuống/2 chiều), tính thông nhau giữa các cầu nối (Sync Direction (Up/Down/Bidirectional link)): khi một cầu nối được thiết lập giữa 2 CSDL thì việc quy định hướng của luồng đồng bộ được xem xét và quyết định tùy thuộc vào yêu cầu về chức năng và quyền hạn của các máy trạm. Ví dụ: trong hệ thống có những máy trạm chỉ muốn sao chép dữ liệu từ máy chủ tập trung xuống chỉ để để xem và thống kê mà không có các thao tác thay đổi (read only) thì hướng đồng bộ chỉ xuống (download only) được áp dụng.

\*Việc chọn giải pháp phù hợp sẽ làm tăng tính an toàn dữ liệu vì việc cập nhật sửa đổi đã được kiểm soát ở mức CSDL (Database level), thấp hơn so với mức ứng dụng (Application level), tính bảo mật dữ liệu cũng được tăng lên.

#### 2.3.2.3. Đụng độ dữ liệu và giải pháp khóa chính GUID

- Đụng độ dữ liệu: trong hệ đồng bộ thì các đụng độ về mặt dữ liệu là không thể tránh khỏi. Đụng độ đặc trưng cho tính không nhất quán trên cấu trúc CSDL (ví dụ: có nhiều hơn 2 khóa chính trùng nhau trong cùng một bảng).

- Đụng độ vật lý trên khóa chính (primary key conflict) [6]: xét 2 CSDL độc lập có cùng cấu trúc bảng và dữ liệu, nếu khóa chính được thiết lập dạng tự động tăng (auto increasement) thì khi gọi phương thức chèn mới (INSERT) trên 2 CSDL, hệ quản trị CSDL địa phương ở cả 2 CSDL trên sẽ có khả năng tạo ra khóa chính trùng nhau rất cao (do tính tuần tự trong cấp phát). Nên khi đồng bộ dữ liệu sẽ bị trùng khóa chính, 1 trong 2 dữ liệu mới đó sẽ bị hệ quản trị loại bỏ.



Sơ đồ 2.14: Đụng độ khóa chính do sử dụng kiểu dữ liệu tự động tăng

=> Sử dụng khóa chính kiểu GUID (Globally Unique Identifier) [6]: GUID là kiểu giá trị "tự nhiên" (natural), được tạo ra dựa trên sự kết hợp giữa các giá trị định danh (địa chỉ MAC của card mạng) và ngẫu nhiên (thời gian hiện tại trên hệ quản trị CSDL), được nhiều hệ thống hỗ trợ. Hệ quản trị CSDL sẽ đảm bảo mỗi GUID được cấp phát là duy nhất trên toàn cầu (mặc dù khả năng trùng là có thể xảy ra trên lý thuyết nhưng thực tế có thể chấp nhận được tùy vào phạm vi sử dụng nội bộ hay liên mạng hay toàn cầu). GUID được chia làm 2 nhóm lớn:

+ GUID ngẫu nhiên: các GUID được tạo ra không theo một trật tự nào cả. Sử dụng trong các trường hợp bảo mật cao (do rất khó để đoán được giá trị cấp phát tiếp theo). Tuy nhiên CSDL lưu trữ GUID dạng này sẽ bị phân mảnh rất nhiều.

+ GUID tuần tự: các GUID được tạo ra theo một trật tự. Giảm sự phân mảnh.

- Đụng độ logic do sự trễ (Delay) dữ liệu: một sự trễ dữ liệu được định nghĩa khi mà sự thay đổi về mặt CSDL ở một máy trạm khác (dù đã đẩy hay chưa đẩy lên máy chủ tập trung) chưa kịp cập nhật cho máy trạm địa phương (local machine) mà máy trạm địa phương cũng đã tạo nên một sự sửa đổi. Khi mà đụng độ về khóa chính (đụng độ vật lý) đã được giải quyết thì đụng độ logic do sự chậm trễ trong việc cập nhật dữ liệu lên máy chủ tập trung là vấn đề không thể tránh khỏi do đặc thù làm việc khi không có mạng (offline), và Sync Framework chỉ có thể can thiệp và xử lý tự động khi hạng mục dữ liệu bị thay đổi ở cả 2 nguồn là không giao nhau hoặc hạng mục dữ liệu giao nhau là đơn nhất. Đối với các trường hợp sự thay đổi ở một hạng mục kéo theo sự thay đổi ở các hạng mục liên quan thì Sync Framework không thể can thiệp được.



Sơ đồ 2.15: Minh họa giải quyết đụng độ khi sử dụng khóa chính là GUID

#### 2.3.2.4. Kỹ thuật khóa Semaphore trong xử lý đa luồng (multi thread) trên hàng đợi (queue)

- Tác vụ bất đồng bộ (asynchronous action): là các tác vụ được gọi chạy nền bằng các luồng riêng biệt mà không cần biết kết quả trả về (cách hoạt động gần giống giao thức UDP trong truyền tin không xác báo).

- Khóa semaphore giúp ngăn chặn (trong semaphore đơn tiến trình) hoặc giảm thiểu (trong semaphore đa tiến trình) sự đụng độ trong truy cập các tài nguyên chia sẻ từ các tác vụ bất đồng bộ (tạo nên bởi các luồng xử lý song song) bằng cách tạo ra các chính sách về phiên và các giao tác trên hàng đợi.



Sơ đồ 2.16: Minh họa semaphore đơn tiến trình trong xử lý đa luồng trên hàng đợi

- Có thể không cần quản lý các luồng truy xuất dữ liệu song song, vì bản thân hệ quản trị CSDL đã có các chính sách về khóa trên dữ liệu, tuy nhiên lúc này sẽ gây tải lên cho cả hệ quản trị CSDL địa phương lẫn máy chủ tập trung. Chúng ta nên có giải pháp quản lý ở mức ứng dụng nhằm tránh gây ra các tải không cần thiết lên máy chủ, nhất là trong môi trường liên mạng (tiết kiệm được tài nguyên hệ thống, ví dụ: băng thông đường truyền, thời gian truyền tải).

- Nhằm đảm bảo dữ liệu có tính sẵn sàng cao và nhất quán, các chính sách sau cần được áp dụng trong kỹ thuật xử lý đồng bộ:

+ Luôn đồng bộ dữ liệu mới nhất từ máy chủ tập trung về máy trạm trước khi thực hiện các sửa đổi trên CSDL (nhằm hạn chế sự trễ dữ liệu).

+ Sau khi một máy trạm hoàn tất một thao tác sửa đổi trên CSDL thì phải đồng bộ lên máy chủ tập trung ngay lập tức (nhằm đảm bảo hệ thống luôn phản ánh đúng CSDL hiện tại).

### 2.3.3. Công nghệ DevExpress trong lập trình giao diện

#### 2.3.3.1. Tổng quan

- Định nghĩa: DevExpress là một Framework được viết cho nền tảng .NET Framework. Cung cấp các control và công nghệ để phục vụ cho quá trình phát triển phần mềm [4].



Hình 2.2: Logo DevExpress

- Thành phần của DevExpress gồm có:

+ WinForm Controls: cung cấp các control cho WinForm.

+ ASP.NET Controls: cung cấp các control cho WebForm.

+ WPF Controls: cung cấp các control cho WPF.

+ Silverlight Controls: cung cấp các control cho Silverlight.

+ XtraCharts: control cung cấp các loại biểu đồ.

+ XtraReports: cung cấp các control tạo báo cáo.

+ XPO: cung cấp môi trường làm việc với database.

+ XAF: một công nghệ mới giúp việc phát triển phần mềm một cách nhanh chóng.

\* Phần mềm chỉ sử dụng ba controls của Devexpress (WinForm, ASP.NET và XtraReports).



Hình 2.3: Các thành phần của DevExpress

- Tính năng: trải qua hàng loạt phiên bản, DevExpress đã từng bước được nâng cấp, hoàn thiện và thêm mới rất nhiều chức năng. Với phiên bản DevExpress 13.2.9 hiện tại, DevExpress đã cung cấp những công cụ, môi trường tuyệt vời để biến những ý tưởng của người lập trình thành hiện thực một cách nhanh chóng, dễ dàng.

+ Sử dụng công cụ của DevExpress giúp cho việc thiết kế giao diện hay chức năng không phải tốn nhiều thời gian, ngoài ra nó giảm được khả năng gây lỗi khi sử dụng công cụ của DevExpress.

- Phiên bản mới nhất: 14.2

- Phiên bản sử dụng trong đề tài: 13.2.9

#### 2.3.3.2. DevExpress dành cho ASP.NET WebForm

- Định nghĩa: DevExpress ASP.NET là một mảng của DevExpress cung cấp các control phục vụ cho Web (bao gồm cả Web Forms và MVC (mã: Razor hoặc ASP)). Sử dụng control của Devexpress ASP.NET làm cho phần mềm thêm sinh động và chuyên nghiệp hơn.

- Các controls mà website sử dụng từ DevExpress:

1. ASPxTabControl



Hình 2.4: Minh họa giao diện ASPxTabControl

- Bộ ASPxTabControl bao gồm 2 thành phần nhỏ, giúp ta tạo nên các Tab cho trang web của mình. Có thể dùng thành phần APSxTabControl chỉ để thể hiện các Tab hoặc dùng ASPxPageControl để tạo các Tab cùng với nội dung bên trong của từng Tab. Cả 2 thành phần này đều hỗ trợ AJAX [1, tr.33-37].

- Đặc điểm:

+ Các Templates có thể xác định cho từng tab trong cả trạng thái kích hoạt hay không kích hoạt.

+ Có thể thay đổi giao diện, sự thể hiện của từng thành phần một cách trực tiếp qua các thuộc tính hoặc qua CSS.

+ Nhiều định dạng phong phú, dễ dàng chọn và thay đổi.

+ Hỗ trợ hai cách để mở một Tab: click chuột hoặc chi rê chuột lên trên Tab.

+ Dễ dàng chỉnh sửa, thay đổi với trình chỉnh sửa thông minh.

+ Có thể tuỳ chỉnh khoảng cách giữa các tab.

+ Dễ dàng tuỳ chỉnh vị trí của các tab.

+ Có thể chèn hình ảnh đại diện cho từng tab.

2. ASPxGridView

- ASPxGridView là một Control rất mạnh trong việc hỗ trợ hiển thị dữ liệu dạng lưới, cho phép ta tạo ra các trường hiển thị bằng tay hoặc lấy từ CSDL [1, tr.58-84].

- Đặc điểm:

+ Tương thích với nhiều trình duyệt.

+ Hỗ trợ AJAX: ta có thể cập nhật nội dung của Control thông qua phương thức callbacks, không cần thiết phải load lại toàn bộ trang.

+ Tương tác với người dùng rất đa dạng.

+ Xuất dữ liệu: hỗ trợ xuất dữ liệu ra định dạng PDF, XLS và RTF.

+ Hỗ trợ SEO (Search Engine Optimization): tối ưu hoá công cụ tìm kiếm.

+ Hỗ trợ khai thác dữ liệu từ nhiều hệ quản trị cơ sở dữ liệu khác nhau: Microsoft Access, SQL Server.

+ Cho phép hiển thị dữ liệu dạng Master - Detail với cấu trúc đa dạng.



Hình 2.5: Minh họa giao diện ASPxGridView

+ Có 2 chế độ chỉnh sửa: từ Form chỉnh sửa hay chỉnh ngay trên hàng.

+ Chứng thực dòng dữ liệu và chỉ ra lỗi: ASPxGridView cho phép ta xác thực bằng tay các dòng đã chỉnh sửa, và hiển thị thông báo lỗi đối với trường không hợp lệ.

+ Tự động gom nhóm dữ liệu: cho phép người dùng gom nhóm dữ liệu, không giới hạn số cột.

+ Tóm tắt dữ liệu đầy đủ: cho phép hiển thị thông tin thống kê như MIN, MAX, AVG, SUM và COUNT trực tiếp trên lưới.

+ Cho phép lọc dữ liệu và hiển thị Text: với mỗi cột ta có thể chỉ định cách dữ liệu được sắp xếp theo giá trị hiển thị của nó. Ngoài ra ta có thể cho phép lọc dữ liệu bất kỳ bằng cách gõ vào giá trị muốn lọc trực tiếp vào ô Textbox.

+ Cho phép lựa chọn nhiều dòng cùng một lúc

- Tuỳ biến giao diện hiển thị:

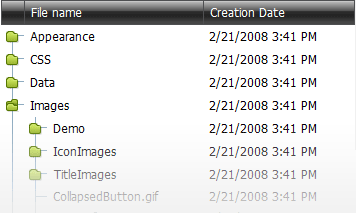
+ Giao diện: ta có thể tuỳ chỉnh giao diện của lưới bằng cách chọn các định dạng hiển thị khác nhau, chỉ sau vài cái click chuột.

+ Hỗ trợ các Template: với mỗi thành phần bên trong ASPxGridView, ta có thể hoàn toàn tuỳ biến việc hiển thị thông qua các control HTML hay bên phía máy chủ.

+ Hỗ trợ CSS đầy đủ.

3. ASPxTreeList

- ASPxTreeList là một Control rất mạnh trong việc hỗ trợ hiển thị dữ liệu dạng cây, cho phép ta tạo ra các trường hiển thị bằng tay hoặc lấy từ CSDL.



Hình 2.6: Minh họa giao diện ASPxTreeList

- Đặc điểm:

+ Tương thích với nhiều trình duyệt.

+ Hỗ trợ AJAX: ta có thể tải danh sách child node của Control thông qua phương thức callbacks, không cần thiết phải tải lại toàn bộ trang.

+ Tương tác với người dùng rất đa dạng.

+ Hỗ trợ khai thác dữ liệu từ nhiều hệ quản trị cơ sở dữ liệu khác nhau: Microsoft Access, MSSQL Server.

+ Theo dõi dòng dữ liệu và chỉ ra lỗi: ASPxTreeList cho phép ta xác thực bằng tay các dòng đã chỉnh sửa, và hiển thị thông báo lỗi đối với trường không hợp lệ.

- Tuỳ biến giao diện hiển thị:

+ Giao diện: có thể tuỳ chỉnh giao diện của cây bằng cách chọn các định dạng hiển thị khác nhau, chỉ sau vài cái click chuột.

+ Hỗ trợ nhiều Template: với mỗi thành phần bên trong ASPxTreeList, ta có thể hoàn toàn tuỳ biến việc hiển thị thông qua các control HTML hay bên phía máy chủ.

+ Hỗ trợ CSS đầy đủ.

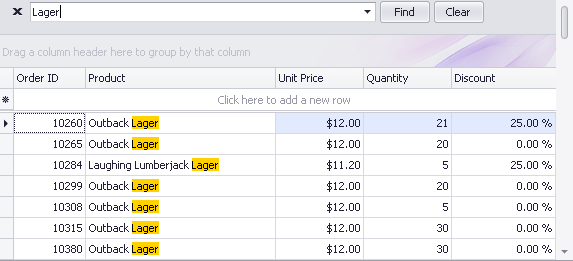
#### 2.3.3.3. DevExpress dành cho .NET WinForm

- Định nghĩa: DevExpress WinForm là một mảng của Devexpress cho phép các Control phục vụ cho WinForm .NET. Sử dụng Control của Devexpress WinForm làm cho phần mềm thêm sinh động và chuyên nghiệp hơn.

- Các Control mà phần mềm sử dụng từ DevExpress:

1. GridControl

- GridControl rất mạnh trong việc hỗ trợ hiển thị dữ liệu dạng lưới, quản lý một lượng lớn dữ liệu, cho phép ta tạo ra các trường hiển thị bằng tay hoặc lấy từ CSDL.



Hình 2.7: Minh họa giao diện GridControl

- Đặc điểm:

+ Chạy Thread riêng biệt để xử lý dữ liệu và hiển thị lên màn hình

+ Giao diện người dùng đa dạng (Banded Grid View, Advanced Banded Grid View, Card View và Layout View).

+ Tương tác với người dùng rất đa dạng.

+ Hỗ trợ khai thác dữ liệu từ nhiều hệ quản trị cơ sở dữ liệu khác nhau: Microsoft Access, MSSQL Server.

+ Cho phép hiển thị dữ liệu dạng Master - Detail với cấu trúc đa dạng.

+ Tự động gom nhóm dữ liệu: cho phép người dùng gom nhóm dữ liệu, không giới hạn số cột.

+ Tóm tắt dữ liệu đầy đủ: cho phép hiển thị thông tin thống kê như MIN, MAX, AVG, SUM và COUNT trực tiếp trên lưới.

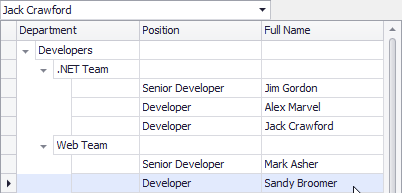
+ Cho phép lọc dữ liệu và hiển thị Text: với mỗi cột ta có thể chỉ định cách dữ liệu được sắp xếp theo giá trị hiển thị của nó. Ngoài ra ta có thể cho phép lọc dữ liệu bất kỳ bằng cách gõ vào giá trị muốn lọc trực tiếp vào ô Textbox.

+ Cho phép lựa chọn nhiều dòng cùng một lúc.

- Tuỳ biến giao diện hiển thị: ta có thể tuỳ chỉnh giao diện của lưới bằng cách chọn các định dạng hiển thị khác nhau, chỉ sau vài cái click chuột.

2. TreeListLookUpEdit

- TreeListLookUpEdit là một Control hỗ trợ chọn dữ liệu từ TreeList, như một Combobox nâng cao.



Hình 2.8: Minh họa giao diện TreeListLookUpEdit

- Đặc điểm:

+ Hiển thị dạng lồng dữ liệu, không giới hạn số nhánh con.

+ Hỗ trợ chức năng lọc, tìm kiếm dữ liệu trong cây.

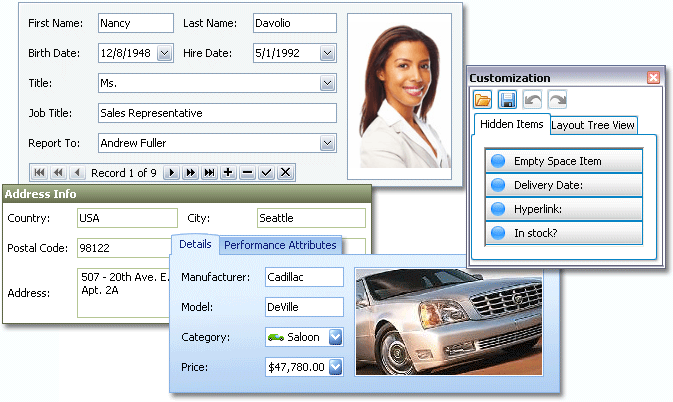
+ Tương tác với người dùng rất đa dạng.

+ Hỗ trợ khai thác dữ liệu từ nhiều hệ quản trị cơ sở dữ liệu khác nhau: Microsoft Access, MSSQL Server

- Tuỳ biến giao diện hiển thị: ta có thể tuỳ chỉnh giao diện của cây bằng cách chọn các định dạng hiển thị khác nhau, chỉ sau vài cái click chuột.

3. Look & Feel

- Định nghĩa: Look & Feel là một chức năng của Devexpress cho phép thay đổi giao diện phần mềm với kho giao diện có sẵn hoặc tự thiết kế.



Hình 2.9: Minh họa giao diện Look & Feel

- Tính năng: cho phép thay đổi giao diện một cách trực quan mà không cần phải restart lại phần mềm.

4. XtraReport

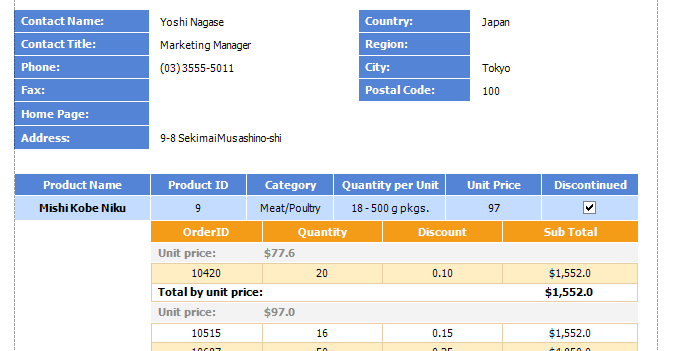
- Devexpress XtraReport là một Control của Devexpress cho phép xuất báo cáo, in ấn dữ liệu ra nhiều loại tập tin khác nhau, cho phép xuất báo cáo dạng động. Khẩu lệnh của DevExpress là: “WYSIWYG”. Sử dụng DevExpress XtraReport làm cho việc xuất báo cáo dễ hơn bao giờ hết [1, tr.84-102].

- Đặc điểm: Devexpress XtraReport chứa công cụ để tạo Report từ dữ liệu phức hợp, chế độ xem trước, in và xuất report ra nhiều định dạng khác nhau.

- Tính năng:

+ XtraReport hoạt động được trong cả ứng dụng Winform và Web Forms. Ta có thể chỉ cần tạo một Report và sử dụng ở 2 môi trường khác nhau.

+ Tích hợp đầy đủ Visual Studio .NET: Report Designer tích hợp. Hỗ trợ chế độ xem trước dạng Report Viewer (Winform) hay HTML (Web Forms) và dạng in, khi có thay đổi trong thiết kế report, ta không cần biên dịch lại toàn bộ ứng dụng mà vẫn có thể cập nhật được chế độ xem trước kịp thời.



Hình 2.10: Minh họa giao diện XtraReport

+ XtraReports làm việc với toàn bộ đối tượng dữ liệu được hỗ trợ bởi Visual Studio .NET như : chuẩn .NET Data Objects, Ilist Interface, XML Data Objects.

+ Cho phép lọc dữ liệu dưới với nhiều cấp: Data adapter, Data set, Data views.

+ Cho phép gom nhóm dữ liệu: gom nhóm đa tầng và lồng nhau.

+ Hỗ trợ nhiều control chuẩn như: Label, Line, BarCode, CheckBox, PageInfo, Panel, PictureBox, PageBreak, Table, ZipCode, ...

+ Hỗ trợ biểu đồ thông qua control XtraCharts.

+ SubReports: có thể dùng lại các lớp của XtraReport vào ứng dụng qua control Subreport. Chỉ cần thả vào control Subreport, set thuộc tính nguồn Report, ta có 2 report từ một nguồn.

+ Hỗ trợ tóm tắt: dễ dàng tạo tóm tắt cho một textbox hay một ô trong bảng. Chỉ cần set 2 thuộc tính Summary position (group hay report) và Summary type (avg, min, max, sum, count…)

+ Hỗ trợ phong phú các định dạng xuất ra: PDF, HTML, MHT, RTF, TXT, CSV và MS Excel. Có thể xuất báo cáo ra định dạng hình ảnh như: BMP, EMF, GIF, JPEG, PNG, TIFF, WMF.

+ Importing: có thể nạp lại report cũ của mình tử MS Access, Crystal Reports, Data Dynamics Active Reports vào XtraReport.

+ Tìm kiếm ở chế độ xem trước: giúp cho người dùng có thể tìm những đoạn text mong muốn.

+ Hỗ trợ kế thừa, Bookmark, Watermarks.

5. Tính năng Static Website Loading trong Web Forms thông qua AJAX

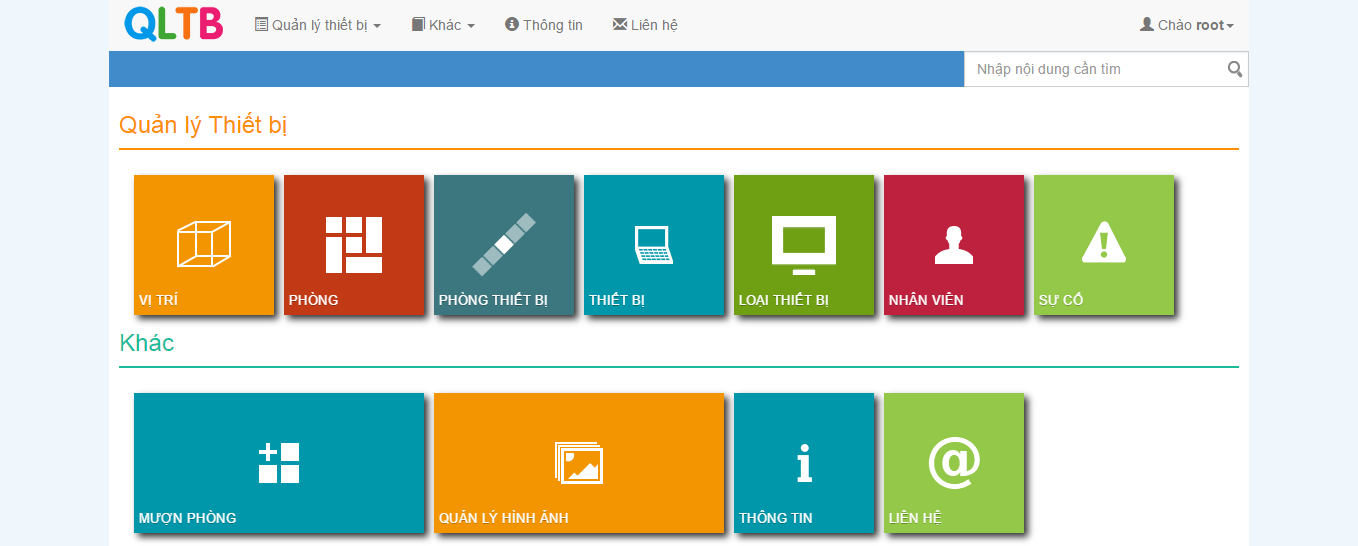
- Về AJAX: Ajax được giới thiệu lần đầu tiên vào ngày 18/02/2005 trong một bài báo có tên "AJAX : A New Approach to Web Applications của tác giả Jesse James Garrett", công ty AdapativePath. Ngay sau đó thuật ngữ AJAX được phổ biến cực kỳ nhanh chóng trong cộng đồng phát triển Web và cho đến nay nó là một trong những từ khóa được tìm kiếm nhiều nhất trên Internet.

- UpdatePanel của Visual Studio hỗ trợ người lập trình web thao tác với ajax một cách dễ dàng thông qua kéo thả, chỉ cần kéo UpdatePanel vào trang aspx, mọi xử lý viết trong thẻ UpdatePanel này sẽ được Visual Studio tự động chuyển sang xử lý dạng Ajax.

### 2.3.4. Công nghệ giao diện tùy biến (responsive design) dành cho ứng dụng Web

#### 2.3.4.1. Công nghệ Responsive Web Design (RWD)

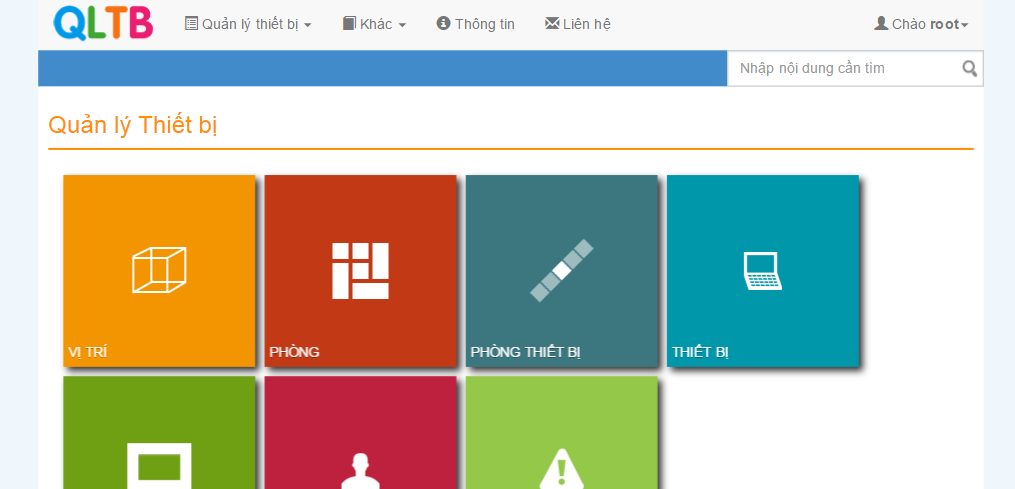
- Định nghĩa: RWD áp dụng nhiều bố cục trang web cho các loại kích cỡ màn hình khác nhau chứ không chỉ thiết kế một giao diện cố định như vẫn thường làm trước đó. Cộng với sự phát triển của các chuẩn HTML5 và CSS3, RWD đã trở thành một thứ quan trọng mà quản trị viên hay chủ sỡ hữu website cần phải nghĩ tới trong bối cảnh ngày càng nhiều thiết bị di động với đủ các kích cỡ, đủ loại độ phân giải màn hình khác nhau được tung ra thị trường.



Hình 2.11: Minh họa giao diện RWD trên Laptop (1366 x 768)

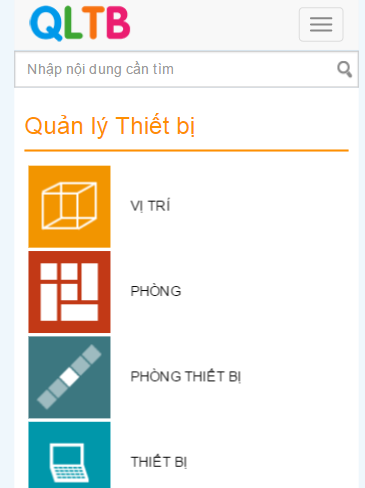
- Lợi ích:

+ RWD dùng để bố cục lại giao diện trang web cho tương thích với nhiều loại kích cỡ màn hình khác nhau.



Hình 2.12: Minh họa giao diện RWD trên iPad 3 (1024 x 768)

+ Trang web có thể hiển thị một cách đầy đủ trên màn hình di động. Tuy nhiên, nếu không áp dụng RWD, trang web khi xem trên thiết bị di động sẽ trở nên nhỏ, bắt buộc phải zoom vào và kéo qua lại, lên xuống liên tục để đọc nội dung. Đây là trải nghiệm tiêu cực, không phù hợp tính tiện ích cho người dùng.



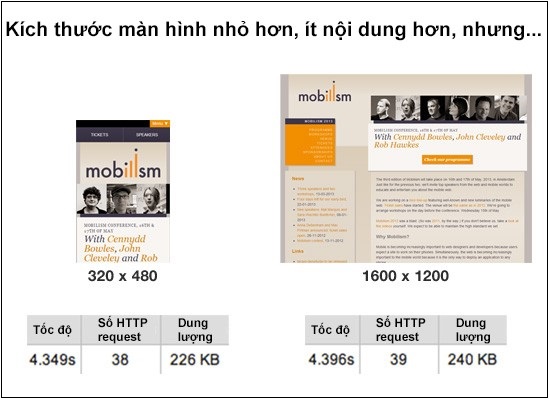
Hình 2.13: Minh họa giao diện RWD trên iPhone 6 (1024 x 768)

+ Tóm lại, RWD là một xu hướng thiết kế hoàn toàn có lợi bởi nó đảm bảo bạn sẽ luôn luôn có những trải nghiệm tốt nhất, đẹp nhất khi xem trang web dù bạn có đang dùng thiết bị nào đi nữa. Nó giúp nhà lập trình web tận dụng tối đa không gian để trình diễn những nội dung cho chúng ta xem theo cách thoải mái và thích thú nhất có thể.

- Hiệu năng:

+ Năm 2009, Guy Podjarny, trưởng nhóm kiến trúc sản phẩm của công ty Akamai (một đơn vị chuyên nghiên cứu và tối ưu hóa tốc độ cho các giải pháp điện toán) đã thử nghiệm 347 trang web RWD được trình diễn trên http://mediaqueri.es bằng cách dùng Chrome trên nhiều thiết bị khác nhau, cộng với công cụ WebPageTest. Podjarny nhận thấy rằng kích thước tải về cũng như tốc độ load của các trang web này không khác mấy khi sử dụng trên nhiều thiết bị khác nhau và màn hình với độ phân giải khác nhau. Sau đó Podjarny đã trình bày phát hiện của mình tại một hội nghị chuyên về thiết kế.

+ Vậy thử nghiệm trên có ý nghĩa gì ? Nó cho thấy rằng mặc dù trang web đã phản hồi lại với kích thước màn hình bằng cách thu gọn nội dung, ẩn bớt những thành phần không cần thiết, thu nhỏ cỡ ảnh... nhưng điều đó không đồng nghĩa với chuyện Web sẽ tải nhanh hơn. Nó cũng không đảm bảo rằng dung lượng tải về sẽ nhỏ hơn, ít chiếm băng thông hơn. Việc tối ưu hóa này hoàn toàn nằm trong tay lập trình viên và nhà thiết kế Web.



Hình 2.14: So sánh tốc độ tải trang Web giữa giao diện Mobile và Desktop

(Nguồn: Tinh tế - www.tinhte.vn [2])

- Bên dưới là biểu đồ so sánh về kích thước trang của các web RWD trên hai loại độ phân giải do Akamai thực hiện. Đến 86% trang web khi tải giữa hai loại màn hình không cho thấy sự khác biệt rõ rệt về dung lượng trang, tức là chúng ta chỉ tiết kiệm được một khoảng dung lượng không đáng kể khi duyệt web bằng mobile.



Sơ đồ 2.17: So sánh giữa kích thước trang Web và độ phân giải màn hình

(Nguồn: Tinh tế - www.tinhte.vn [2])

- Độ phức tạp:

+ Thật ra trước đây các lập trình viên cũng có nghĩ đến một biện pháp khác khi mà RWD chưa phổ biến, đó là xây dựng một phiên bản di động dành cho web (ví dụ: m.tinhte.vn). Cách này cũng tốt, tác dụng gần như tương đương với RWD. Tuy nhiên, nó là một trang tập tin HTML riêng, một file CSS riêng được viết riêng cho thiết bị di động, hình ảnh cũng được thiết lập với kích thước nhỏ hơn.

+ Ngoài ra, một số dịch vụ Online cũng có hỗ trợ chuyển RSS thành một trang web riêng biệt. Lập trình viên có thể tận dụng điều này để thiết kế web cho thiết bị di động mà không phải đầu tư quá nhiều công sức. Họ chỉ cần làm cho RSS của mình đầy đủ nhất có thể là xong. Người dùng truy cập từ các smartphone, tablet sẽ không thấy giao diện chính mà chỉ thấy các dòng cập nhật mới nhất, tin tức mới nhất. Như vậy cũng đã đủ đối với một số trang web rồi.

+ RWD thì ngược lại, nó vốn dĩ là phức tạp hơn bởi nhà thiết kế web đang cố gắng nhiều trải nghiệm xem khác nhau chứ không nhắm đến một loại thiết bị cụ thể nào cả. Điều đó có nghĩa là trình duyệt trên máy mobile phải đảm đương một file HTML lớn, một tập tin CSS cũng lớn không kém. Nếu không được tích hợp tốt, RWD có thể làm cho việc duyệt web di động trở nên chậm chạp hơn mặc dù bố cục rất tốt.

#### 2.3.4.2. CSS Framework

- Định nghĩa: là bộ công cụ giúp thiết kế trang Web bằng CSS nhanh hơn. Nghĩa là nó được trừu tượng hóa lên một mức cao hơn. Thay vì phải hiểu rõ về các bộ chọn, các thuộc tính và giá trị trong CSS để style cho trang web của mình, thì chỉ cần biết các thành phần có trên trang Web như: Form, Navbar, Tooltip, Dropdown- Menu, Modal, Button,... và thêm nó vào trang HTML một cách thích hợp. Công việc còn lại là của CSS Framework.

- Phiên bản: hiện tại có rất nhiều CSS Framework được phát triển. Trong đó, có 2 loại được phát triển mạnh nhất đến thời điểm này là:

+ Bootstrap (2 và 3)

+ Foundation (3, 4 và 5)

- Tính năng:

+ Hỗ trợ khả năng Responsive: tức là trang web sẽ tự động co giãn theo kích thước của cửa sổ trình duyệt.

+ Tương thích tốt với thiết bị cỡ nhỏ: với sự phổ biến của thiết bị thông minh như hiện nay, đây là một yếu tố quan trọng. Không cần phải thiết kế một bản riêng cho mobile, với bootstrap bạn chỉ cần thiết kế một lần cho mọi thiết bị.

+ Được tích hợp với thư viện jQuery và tương tác tốt với chuẩn HTML5 và CSS3.

\*Twitter Bootstrap (gọi tắt là Bootstrap).

- Đặc điểm:

+ Bootstrap là một CSS Framework phổ biến nhất hiện nay do Twitter phát triển.

+ Bootstrap bao gồm các mã CSS + HTML cơ bản cho Typography, Forms, Buttons, Tables, Grids, Navigation, và nhiều thành phần khác của Web.

+ Twitter Bootstrap cung cấp lưới cố định (fixed) rộng 940px và 12 cột. Tất nhiên là cũng có giải pháp cho việc dùng layout dạng động (fluid).

+ Style của các phần tử HTML trong Twitter Bootstrap khá đơn giản và thanh lịch. Ví dụ như phần đổ bóng trong input, highlight của bảng biểu, các mã CSS hiển thị cảnh báo, tabs, phân trang…

- Tính năng: Bootstrap giúp chúng ta giảm thiểu thời gian thiết kết html và css. Bootstrap định nghĩa sẳn các class css. Công việc của chúng ta chỉ là sử dụng các class đó vào mục đích của mình. Bootstrap còn hổ trợ RWD, một cách làm giao diện đa thiết bị rất được ưu chuộng trong thời gian gần đây.

- Phiên bản mới nhất: 3

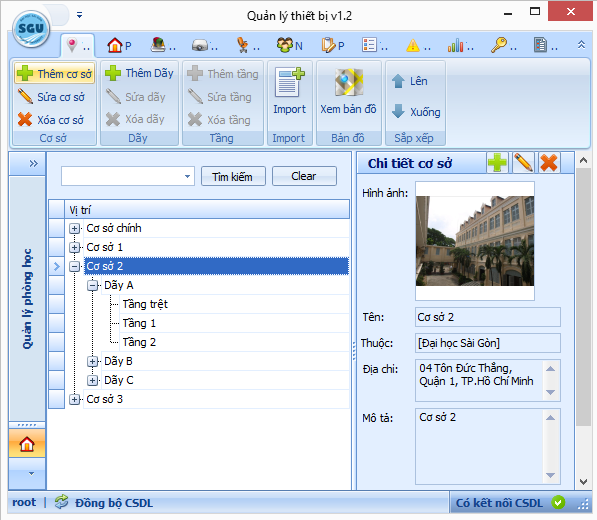
- Phiên bản sử dụng trong ứng dụng: 3

## 2.4. Kết quả thực thi (các màn hình chức năng)

- Do số lượng màn hình chức năng nhiều nên chỉ liệt kê đại diện vài chức năng chính.

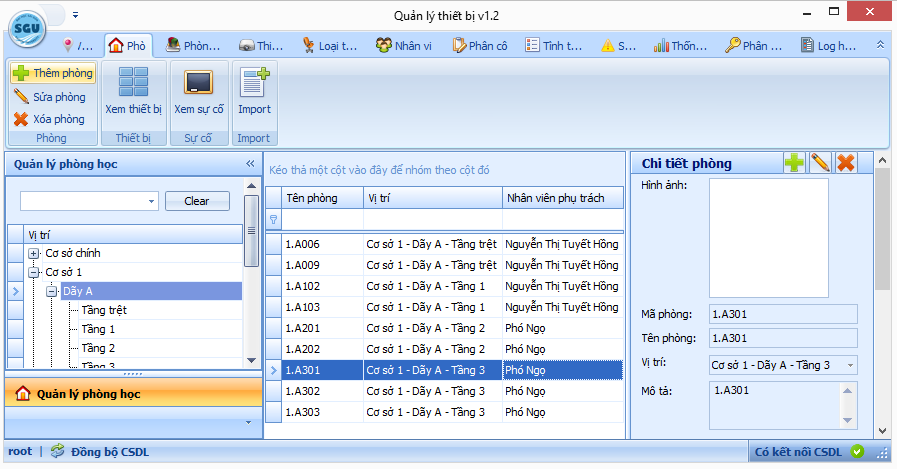
- Ứng dụng Windows Desktop:

+ Chức năng quản lý vị trí: cơ sở, dãy, tầng.



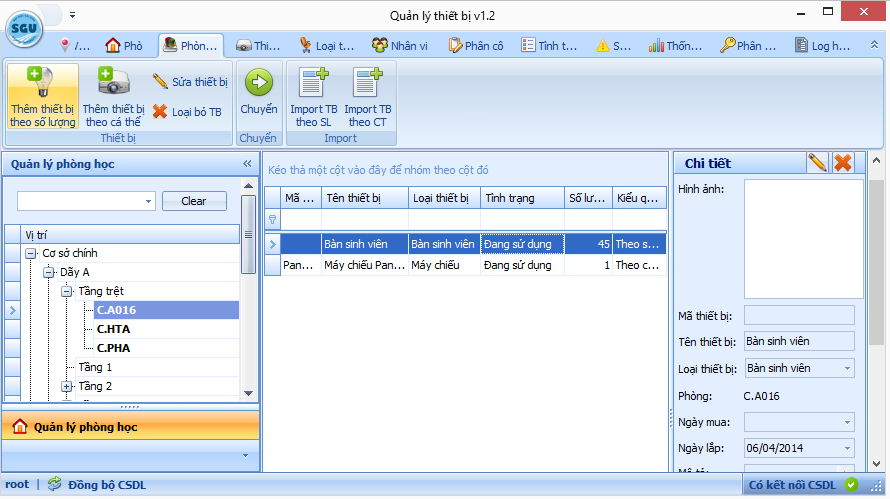
Hình 2.15: Màn hình chức năng quản lý vị trí (cơ sở, dãy, tầng)

+ Chức năng quản lý phòng



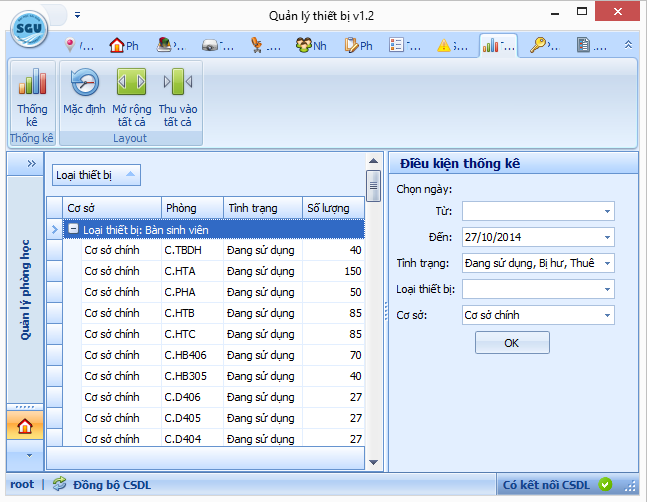
Hình 2.16: Màn hình chức năng quản lý phòng

+ Chức năng quản lý thiết bị



Hình 2.17: Màn hình chức năng quản lý thiết bị

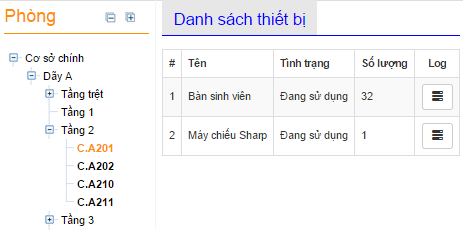
+ Chức năng thống kê thiết bị



Hình 2.18: Màn hình chức năng thống kê thiết bị

- Ứng dụng Windows Web:

+ Chức năng xem thiết bị theo phòng



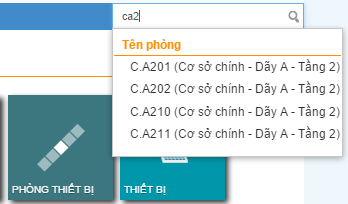
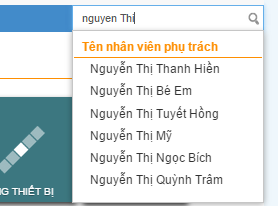
Hình 2.19: Màn hình chức năng xem thiết bị theo phòng (Web)

+ Chức năng xem danh sách, thông tin nhân viên phụ trách



Hình 2.20: Màn hình chức năng xem danh sách nhân viên phụ trách (Web)

+ Chức năng tìm kiếm

Hình 2.21: Màn hình chức năng tìm kiếm (Web)

CHƯƠNG 3: KIỂM THỬ VÀ TRIỂN KHAI

## 3.1. Kiểm thử tự động mức mã nguồn (Unit test)

### 3.1.1. Kiểm thử hộp đen (Black box testing)

- Trong Unit test có 2 phương pháp kiểm thử cơ bản là kiểm thử hộp trắng (kiểm thử luôn cả dữ liệu vào/ra và cả cách thực hiện cụ thể) và kiểm thử hộp đen (chỉ kiểm thử các bộ dữ liệu vào/ra mà không quan tâm đến cách thực hiện cụ thể).

- Do quy mô của ứng dụng không quá phức tạp nên trong phạm vi đề tài này chỉ chọn giải pháp kiểm thử hộp đen.

### 3.1.2. Mô hình kiểm thử AAA (Arrange - Act - Assert)

- Định nghĩa: đây là cách tổ chức kiểm thử Unit test phổ biến nhất. Trong đó:

+ Arrange (sắp xếp): là lựa chọn các thành phần tham gia kiểm thử. có thể là các biến, các lớp hoặc thậm chí là các dự án ngoài.

+ Act (Action - hiện thực): là cài đặt các bước, thao tác trong lịch trình kiểm thử nhằm tạo ra các kết quả đầu ra (output) tương ứng với các kết quả đầu vào (input).

+ Assert (đánh giá): là một so sánh giữa kết quả đầu ra thực tế và kết quả đầu ra mong muốn. Từ đó đưa ra đánh giá cuối cùng.

## 3.2. Kiểm thử chấp nhận (Acceptance test)

- Kiểm thử chấp nhận được thực hiện bởi người dùng hoặc lập trình viên đứng ở góc độ là người dùng.

- Giai đoạn kiểm thử này có thể được thực hiện thủ công hoặc tự động tùy thuộc vào người đứng đầu dự án kiểm thử. Do đặc tính của kiểm thử tự động đòi hỏi phải bỏ ra nhiều thời gian cho phần thiết kế hệ thống hơn là thời gian thực thi kiểm thử, nên nhóm đã quyết định chọn giải pháp kiểm thử thủ công cho giai đoạn này.

- Do lịch trình kiểm thử rất nhiều nên sau đây chỉ liệt kê một vài chức năng đại diện.

Bảng 3.1: Kết quả kiểm thử chấp nhận

Trang 71

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Step** | | **Type** | **Description** | **Data** | **Expected result** | **Actual result** | **Pass or Fail** | **Note** |
| **Thêm 1 thiết bị quản lý theo cá thể** | | | | | | | | |
| 1 | Step | | Mở chương trình quản lý thiết bị |  |  |  |  |  |
| 2 | Step | | Click vào ribbon thiết bị |  | Ribbon thiết bị hiện lên và load dữ liệu cần thiết |  |  |  |
| 3 | Step | | Click vào tab "Thiết bị quản lý theo cá thể" |  | Những thiết bị quản lý theo cá thể được hiện ra trên gridcontrol |  |  |  |
| 4 | Step | | Click vào button "Thêm thiết bị" trên ribbon |  | Form thêm thiết bị được enable, tiêu đề đổi thành "Thêm thiết bị" Các Button thêm, sửa xóa thiết bị phải bị disable |  |  |  |
| 5 | Step | | Click vào Button "Chọn" để chọn ảnh cho thiết bị |  | Form quản lý hình ảnh hiện ra |  |  |  |
| 6 | Verify Point | | Kiểm tra xem Form quản lí hình ảnh có hiện ra |  |  |  |  |  |
| 7 | Step | | Click vào Button "Thêm mới" |  |  |  |  |  |
| 8 | Verify Point | | Openfile Dialog hiện ra để chọn file ảnh cần upload |  |  |  |  |  |
| 9 | Step | | Chọn 1 file ảnh đuôi JPG |  |  |  |  |  |
| 10 | Verify Point | | Form quản lí ảnh phải up ảnh lên HOST và hiển thị ảnh đã chọn |  |  |  |  | Thời gian upload ảnh không quan trọng  Trang 69 |
| 11 | Step | | Click vào Button "Ok" trên Form quản lí ảnh |  |  |  |  |  |
| 12 | Verify Point | | Các ảnh đã chọn phải được hiển thị trên ImageSlider |  |  |  |  |  |
| 13 | Step | | Điền dữ liệu cho Form thêm thiết bị | Field "Mã thiết bị": "TB001" Field "Tên thiết bị": "Máy chiếu Sony" Field "Loại thiết bị": chọn "Máy chiếu" Field "Ngày mua": "01/10/2014" Field "Mô tả": "Máy chiếu Sony" |  |  |  |  |
| 14 | Step | | Click vào Button "Ok" trên Form thêm thiết bị |  | Hiện thông báo thêm thiết bị thành công Thiết bị mới được thêm vào và hiển thị trên GridControl Form thêm thiết bị bị ẩn đi Các Button chức năng hiển thị lại bình thường Tiêu đề đổi lại là "Chi tiết" Form thông tin hiển thị đối tượng đang được focus trong GridControl (nếu không có thì xóa sạch form) | Như bên expected result |  |  |
| **Kết quả** | | | | | | | **Pass** |  |
| **Chuyển 1 thiết bị được quản lý theo cá thể vào 1 phòng nào đó** | | | | | | | | |
| 1 | Step | | Mở chương trình quản lý thiết bị |  |  |  |  |  |
| 2 | Step | | Click vào ribbon phòng - thiết bị |  | Ribbon phòng - thiết bị hiện lên và load dữ liệu cần thiết |  |  |  |
| 3 | Step | | Tại TreeList vị trí chọn "Cơ sở chính" - > "Dãy A" - > "Tầng 2" - > "C.A201" |  | Những thiết bị của phòng C.A201 được hiện lên GridControl Button "Thêm thiết bi theo số lượng" và button "Thêm thiết bị theo cá thể" được enable |  |  |  |
| 4 | Step | | Click vào Button "Thêm thiết bị theo cá thể" |  | Form thêm thiết bị theo cá thể cho phòng được hiện lên Những thiết bị theo cá thể chưa thuộc phòng nào được hiện lên trên GridControl |  |  |  |
| 5 | Step | | Chọn máy chiếu có mã thiết bị là "TB001" |  | Thông tin của máy chiếu có mã thiết bị là "TB001" được hiện trên form thông tin |  |  |  |
| 6 | Step | | Click vào Button "Thêm và đóng" |  | Form chuyển tình trạng cho thiết bị được hiện lên |  |  |  |
| 7 | Step | | Chọn trạng thái cho máy chiếu này là "Đang sử dụng" |  |  |  |  |  |
| 8 | Step | | Click vào Button "Ok" |  | 2 Form chuyển tình trạng cho thiết bị và form thêm thiết bị theo cá thể được đóng |  |  |  |
| 9 | Verify Point | | Kiểm tra xem máy chiếu được chọn có hiện trên GridControl những thiết bị thuộc phòng không |  | Máy chiếu được chọn được hiển thị Thiết bị đang được focus trên GridControl sẽ được hiển thị trên Form thông tin | Như bên expected result |  |  |
| **Kết quả** | | | | | | | **Pass** |  |
| **Phân công phòng cho nhân viên phụ trách** | | | | | | | | |
| 1 | Step | | Mở chương trình quản lý thiết bị |  |  |  |  |  |
| 2 | Step | | Click vào ribbon nhân viên |  | Ribbon nhân viên hiện lên và load dữ liệu cần thiết |  |  |  |
| 3 | Step | | Chọn nhân viên có tên là "Phạm Thị Mây" trên gridcontrol |  | Thông tin của nhân viên tên "Phạm Thị Mây" được hiện trên Form thông tin |  |  |  |
| 4 | Step | | Click vào Button "Phân công" trên ribbon |  | GridControl hiện lên cây vị trí (có phòng) Có đánh dấu những phòng mà nhân viên "Phạm Thị Mây" đã được phân công |  |  |  |
| 5 | Step | | Tìm phòng C.A501 và đánh dấu phòng đó |  | Danh sách phân công của nhân viên "Phạm Thị Mây" thêm phòng C.A501 vào |  |  | Đây chỉ là thêm tạm, chưa đưa vào cơ sở dữ liệu  Trang 72 |
| 6 | Step | | Click vào Button "Ok" ở bên Form thông tin |  | Hiện thông báo phân công nhận viên thành công |  |  |  |
| 7 | Verify Point | | Chọn lại nhân viên tên "Phạm Thị Mây", xem danh sách phòng được phân công có thêm phòng C.A501 vào hay không |  | Danh sách phòng được phân công sẽ có C.A501 | Như bên expected result |  |  |
| **Kết quả** | | | | | | | **Pass** |  |
| **Tìm kiếm một phòng trên web** | | | | | | | | |
| 1 | Step | | Mở webbrowser lên, nhập url là "http://tb.dsesgu.edu.vn/" |  | Webbrowser hiện trang web của quản lý thiết bị |  |  | Webbrowser sử dụng ở đây là google chrome |
| 2 | Step | | Ô tìm kiếm gõ C.A501 |  |  |  |  | Phòng tìm kiếm là C.A501  Trang 73 |
| 3 | Verify Point | | Kết quả tìm kiếm được hiện dưới dạng dropdown list, nếu không tìm thấy thì hiện thông báo là "Không tìm thấy" |  | Nếu tìm thấy phòng C.A501 thì phải hiện lên thông tin của A.C501 (vị trí của nó) Ngoài ra phải hiện C.A501 là kết quả tìm kiếm của hạng mục nào (trường hợp ở đây là tên phòng) |  |  |  |
| 4 | Step | | Click vào dòng "C.A501 (Cơ sở chính - Dãy A - Tầng 5)" |  |  |  |  | Trường hợp này là tìm thấy khóa "C.A501" |
| 5 | Verify Point | | Web Browser được chuyển sang 1 trang khác, trang này là thông tin của phòng C.A501 |  |  |  |  |  |
| 6 | Verify Point | | Thông tin của phòng C.A501 được hiện lên (gồm thông tin phòng và nhân viên phụ trách) TreeList vị trí trỏ vào vị trí của phòng C.A501 |  |  |  |  |  |
| **Kết quả** | | | | | | | **Pass** | Trang 74 |

Trang 68

Trang 70

Trang 71

## 3.3. Các mô hình triển khai

- Hệ thống khi được đưa vào sử dụng thực tế sẽ có nhiều cách triển khai, tùy thuộc vào điều kiện hạ tầng mạng và yêu cầu chức năng của từng khu vục. Trong đó cách triển khai sau đây là tổng quát nhất, và được hỗ trợ hoàn toàn trong phần mềm.



Sơ đồ 3.1: Mô hình triển khai hệ thống phần mềm

- Cụ thể quá trình triển khai thực tế:

+ Máy chủ: từ nhà cung cấp dịch vụ Web Host VDC.

+ Dung lượng lưu trữ trên Host: 1GB.

+ Băng thông hàng tháng: 20GB.

+ Có tổng cộng 2 CSDL được tạo ra, 1 bản chính thức, 1 bản dự phòng.

+ Máy chủ Web: IIS ASP.NET 4.0.

+ Các máy trạm người dùng đã cài đặt được cho 2 máy tính cá nhân ở ban Hạ tầng Cơ sở, một máy trạm điều khiển và giám sát dữ liệu đặt tại phòng A106 (khoa CNTT).

- Các lưu ý trước khi triển khai hệ thống:

+ Thời gian ở các máy trạm phải là thời gian hiện tại theo múi giờ GMT +7 (Việt Nam).

+ Thời gian ở máy chủ không quan trọng.

+ Default Schema của tất cả User trên CSDL trong MSSQL Server phải thống nhất là "dbo" (tránh tình trạng xung đột phân quyền giữa các trigger trên bảng).

- Các lưu ý sau khi triển khai hệ thống:

+ Mặc dù bản thân hệ thống đã có các biện pháp bảo mật và chính sách để vận hành tốt trong điều kiện bình thường, tuy nhiên khi đưa vào vận hành thực tế thì sẽ không thể tránh khỏi các vấn đề phát sinh liên quan như: máy chủ bảo trì hoặc gặp sự cố dữ liệu, quản trị cấp cao (root) quên mật khẩu, cài lại phần mềm trên máy cá nhân, bị tin tặc tấn công,... Khi đó hệ thống có thể sẽ không làm việc ổn định thậm chí ngừng hoạt động.

=> Cần phải có ít nhất 1 quản trị luôn theo dõi giám sát các bất thường xảy ra trên toàn hệ thống, để từ đó đưa ra giải pháp khắc phục nhanh nhất.

+ Cần chuẩn bị sẵn các phương án dự phòng khi máy chủ hiện tại không làm việc.

+ Sao lưu dữ liệu thường xuyên tùy thuộc vào tính chất của công việc, tần suất thay đổi CSDL.

KẾT LUẬN VÀ KIẾN NGHỊ

**- Phần Kết luận: Phần này nhằm rút ra những kết luận ngắn gọn súc tích về những vấn đề chính mà khoá luận đã trình bày. Kết luận phải tuân thủ nguyên tắc mở để có thể tiếp tục phát triển nghiên cứu, nên tránh các kết luận cứng nhắc, cực đoan.**

*- Nhóm đã vận dụng và kết hợp thành công các công nghệ và kỹ thuật hiện có vào trong ứng dụng. Và đã đạt được kết quả như mục đích đã đề ra ban đầu. Có được điều đó là do chọn được hướng nghiên cứu đúng đắn và phù hợp.*

*- Các kiến thức chỉ giải quyết cụ thể một mảng, lĩnh vực nhất định. Cần phải biết cách so sánh để tìm ra được mối liên hệ, từ đó kết hợp lại với nhau. Như trường hợp ứng dụng đồ thị trong "Lý thuyết đồ thị" vào kỹ thuật xử lý tiến trình đồng bộ. Vấn đề sẽ được giải quyết một cách có khoa học hơn.*

*- Có những lý thuyết dù đã được hệ thống hóa tốt nhưng khi triển khai thực tế lại phát sinh nhiều vấn đề phải giải quyết, thậm chí có những vấn đề chỉ nảy sinh khi bước vào giai đoạn cuối. Điều đó gây ít nhiều khó khăn trong việc sửa lỗi, do các hệ thống con đã kết hợp với nhau.*

*- Tài liệu về các công nghệ như đã trình bày phần lớn là của nước ngoài (tiếng anh là chủ yếu), cũng là một khó khăn cho nhóm khi đào sâu vào các mảng chưa được phổ biến (nhất là các hướng giải quyết chi tiết vấn đề).*

*- Tài liệu trên mạng thì nhiều, tuy nhiên cần phải tổng hợp và chọn lọc lại những gì cần thiết nhất, tránh lan man đi lệch hướng, sẽ gây lãng phí thời gian, chậm tiến độ.*

*- Để có được một bản thiết kế hợp lý, nhóm phải tự đặt ra vấn đề, lấy các ví dụ cụ thể, sau đó tự phản biện để tìm ra được lỗ hổng.*

*- Nhóm rút ra quan điểm: việc nghiên cứu lý thuyết phải đi đôi với thực hành, để thấy được ngay những vướng mắc, những hạn chế,...Từ đó đưa ra hướng mới sớm nhất có thể.*

*- Hãy tin tưởng vào người hướng dẫn, vì họ đã có những kiến thức và thực tiễn sâu rộng. Những chỉ bảo của họ sẽ giúp chúng ta có được định hướng đúng khi giải quyết một vấn đề. Tuy nhiên không quá phụ thuộc và ỷ lại vào người hướng dẫn, điều đó sẽ có hại, vì mục đích khi thực hiện khóa luận là rèn luyện cho sinh viên khả năng tự nghiên cứu và giải quyết vấn đề, người hướng dẫn chỉ định hướng.*

*- Những khó khăn trong suốt quá trình làm việc có thể kể đến như:*

*+ Sự tương thích phiên bản: lúc đầu nhóm chọn phiên bản .NET mới nhất là 4.5 để thực hiện, khi đã vào giai đoạn giữa thì được thông tin nhà cung cấp dịch vụ chỉ hỗ trợ phiên bảo cao nhất là 4.0 (giữa 4.0 và 4.5 là một sự khác biệt lớn, do sự tương thích kéo theo giữa các thành phần khác). Không còn lựa chọn nào khác, buộc lòng phải tìm mọi cách để hạ cấp từ 4.5 về 4.0, nhóm đã phải làm việc xuyên suốt cả một ngày, và rât vui mừng khi đã hoàn thành nhiệm vụ.*

*+ Website ASP.NET: sau khi dữ diệu từ Server trả về cho phía Client, nếu trong thời gian nhất định, phía Client không có phản hồi gì thì DBContext sẽ bị hủy (dispose). Sau thời gian đó nếu Client gửi yêu cầu lên Server sẽ bị lỗi ViewState. Vì thế phải tự động tải lại trang để reset DbContext trên Server.*

*- Lúc ban đầu, được người hướng dẫn giới thiệu triết lý triển khai phần mềm Agile, nhóm đã vận dụng xuyên suốt vào trong toàn bộ quy trình. Và cuối cùng, cũng không biết chắc được là có thực hiện đúng tinh thần Agile hay không ? Tuy nhiên thành quả là có.*

DANH MỤC TÀI LIỆU THAM KHẢO

**Tài liệu tiếng việt**

1. Nguyễn Mai Lĩnh (2010), Hướng dẫn sử dụng bộ công cụ DevExpress cho ASP.NET, Phòng công nghệ thông tin - Đại học Sư phạm TP. Hồ Chí Minh, truy cập ngày 12 tháng 7 năm 2014, <http://123doc.vn/document/1314168-huong-dan-su-dung-bo-cong-cu-devexpress-cho-asp-net.htm>.

2. Tinh tế, Responsive Web Design là gì và nó giúp ích như thế nào cho việc duyệt web trên thiết bị di động, truy cập ngày 10 tháng 08 năm 2014, < https://www.tinhte.vn/threads/responsive-web-design-la-gi-va-no-giup-ich-nhu-the-nao-cho-viec-duyet-web-tren-thiet-bi-di-dong.2101375/>.

**Tài liệu tiếng anh**

3. Entity Framework Tutorial, Database Initialization Strategies in Code - First, truy cập ngày 13 tháng 07 năm 2014, <http://www.entityframeworktutorial.net/code-first/database-initialization-strategy-in-code-first.aspx>.

4. DevExpress, DevExpress Documentation 13.2.9, truy cập ngày 14 tháng 07 năm 2014, <https://documentation.devexpress.com/#HomePage/CustomDocument9453 >.

5. Microsoft MSDN, Entity Framework (EF) Documentation, truy cập ngày 13 tháng 07 năm 2014, <http://msdn.microsoft.com/en- us/data/ee712907.aspx>.

6. Microsoft MSDN, Microsoft Sync Framework, truy cập ngày 13 tháng 07 năm 2014, < http://msdn.microsoft.com/en-us/library/bb902854(v=sql.110).aspx >.

7. Microsoft MSDN, Tutorial: Synchronizing SQL Server and SQL Express, truy cập ngày 12 tháng 07 năm 2014, <http://msdn.microsoft.com/en-us/library/ff928700(v=sql.110).aspx>.

8. Microsoft MSDN, Selecting the Appropriate Sync Framework Components, truy cập ngày 13 tháng 07 năm 2014, <http://msdn.microsoft.com/en-us/library/bb902854(v=sql.110).aspx>.

9. Scott Ambler (2002), Agile Modeling: Effective Practices for eXtreme Programming and the Unified Process, Wiley Computer Publishing - John Wiley & Sons, Inc., New York.

10. Tutorials Point, UML 2.0 - Overview, truy cập ngày 10 tháng 09 năm 2014, <http://www.tutorialspoint.com/uml/uml\_2\_overview.htm>.

11. Vivek Thakur (2008), ASP.NET 3.5 Application Architecture and Design, Packt Publishing Ltd., Birmingham - B27 6PA - UK.

12. Weblogs.asp.net, Inheritance with EF Code First: Part 3 - Table per Concrete Type (TPC), truy cập ngày 13 tháng 07 năm 2014, <http://weblogs.asp.net/manavi/inheritance-mapping-strategies-with-entity-framework-code-first-ctp5-part-3-table-per-concrete-type-tpc-and-choosing-strategy-guidelines>.

PHỤ LỤC A

- Do số lượng bảng nhiều, không thể trình bày hết, nên chỉ trình bày đại diện một vài bảng.

Bảng PL1.1: Sơ đồ bảng COSOS

|  |  |  |
| --- | --- | --- |
| Tên cột | Kiểu dữ liệu | Cho phép rỗng |
| id | uniqueidentifier |  |
| ten | nvarchar(255) |  |
| diachi | nvarchar(MAX) |  |
| subId | nvarchar(MAX) |  |
| [order] | bigint |  |
| mota | nvarchar(MAX) |  |
| date\_create | datetime |  |
| date\_modified | datetime |  |

Bảng PL1.2: Sơ đồ bảng VITRIS

|  |  |  |
| --- | --- | --- |
| Tên cột | Kiểu dữ liệu | Cho phép rỗng |
| id | uniqueidentifier |  |
| coso\_id | uniqueidentifier |  |
| day\_id | uniqueidentifier |  |
| tang\_id | uniqueidentifier |  |
| subId | nvarchar(MAX) |  |
| [order] | bigint |  |
| mota | nvarchar(MAX) |  |
| date\_create | datetime |  |
| date\_modified | datetime |  |

Bảng PL1.3: Sơ đồ bảng PHONGS

|  |  |  |
| --- | --- | --- |
| Tên cột | Kiểu dữ liệu | Cho phép rỗng |
| id | uniqueidentifier |  |
| ten | nvarchar(MAX) |  |
| vitri\_id | uniqueidentifier |  |
| nhanvienpt\_id | uniqueidentifier |  |
| quantrivien\_id | uniqueidentifier |  |
| subId | nvarchar(MAX) |  |
| [order] | bigint |  |
| mota | nvarchar(MAX) |  |
| date\_create | datetime |  |
| date\_modified | datetime |  |

Bảng PL1.4: Sơ đồ bảng THIETBIS

|  |  |  |
| --- | --- | --- |
| Tên cột | Kiểu dữ liệu | Cho phép rỗng |
| id | uniqueidentifier |  |
| ten | nvarchar(MAX) |  |
| ngaymua | datetime |  |
| loaithietbi\_id | uniqueidentifier |  |
| subId | nvarchar(MAX) |  |
| [order] | bigint |  |
| mota | nvarchar(MAX) |  |
| date\_create | datetime |  |
| date\_modified | datetime |  |

Bảng PL1.5: Sơ đồ bảng CTTHIETBIS

|  |  |  |
| --- | --- | --- |
| Tên cột | Kiểu dữ liệu | Cho phép rỗng |
| id | uniqueidentifier |  |
| ngay | datetime |  |
| soluong | int |  |
| phong\_id | uniqueidentifier |  |
| thietbi\_id | uniqueidentifier |  |
| tinhtrang\_id | uniqueidentifier |  |
| subId | nvarchar(MAX) |  |
| [order] | bigint |  |
| mota | nvarchar(MAX) |  |
| date\_create | datetime |  |
| date\_modified | datetime |  |

\*Lược đồ các bảng định nghĩa quan hệ n - n (do sự giống nhau nên chỉ trình bày đại diện 1 sơ đồ cho mỗi loại quan hệ).

- Quan hệ n - n giữa thực thể với hình ảnh:

COSOS\_HINHANH, CTTHIETBI\_HINHANH, DAY\_HINHANH, LOGSUCOPHONG\_HINHANH, LOGTHIETBI\_HINHANH, NHANVIENPT\_HINHANH, PHONG\_HINHANH, SUCOPHONG\_HINHANH, TANG\_HINHANH, THIETBI\_HINHANH.

- Quan hệ n - n giữa thực thể và quyền:

COSOS\_PERMISSION, DAY\_PERMISSION, TANG\_PERMISSION, PHONG\_PERMISSION.

- Quan hệ n - n giữa nhóm quyền và quyền:

GROUP\_PERMISSION

PHỤ LỤC B

\*Ghi chú:

- Kiểu dữ liệu có dấu ? liền sau cho biết kiểu dữ liệu này chấp nhận giá trị null (trong trường hợp kiểu dữ liệu dạng nguyên thủy (primitive) hoặc kiểu cấu trúc (struct), ví dụ kiểu int, long, DateTime,...).

- Kiểu dữ liệu "T" được sử dụng trong kế thừa để lớp cha hiểu và nhận diện đúng lớp kế thừa (parameterized).

- Các thuộc tính kế thừa sẽ không liệt kê lại ở các lớp dẫn xuất để tránh dài dòng (xem chi tiết ở lớp cha).

- Các phương thức:

+ Được liệt kê cuối cùng ở mỗi bảng.

+ Được đặc trưng bởi cặp ngoặc tròn ở sau tên phương thức.

+ Chỉ liệt kê tên phương thức (không liệt kê tham số truyền vào).

+ Chỉ liệt kê các phương thức quan trọng, chính yếu.

1. \_CRUDInterface<T>

- Mức truy cập: public.

- Loại: giao diện.

- Mô tả: Các lớp cần các phương thức thông dụng trên một thực thể: thêm, xóa, sửa,... sẽ thực thi lớp giao diện này.

Bảng PL2.1: Thiết kế giao diện \_CRUDInterface<T>

|  |  |  |
| --- | --- | --- |
| Tên phương thức | Kiểu trả về | Mô tả |
| add() | int | Thêm đối tượng vào CSDL |
| update() | int | Sửa đối tượng trong CSDL |
| delete() | int | Xóa đối tượng khỏi CSDL |
| reload() | T | Tải lại đối tượng |
| trigger() | void | Ép lazyloading tải dữ liệu ngay |
| doTrigger() | void | Gọi trigger trên tât cả các thuộc tính là đối tượng đơn và là khóa ngoại |
| clone() | T | Sao chép thông tin ra đối tượng mới |
| moveUp() | void | Di chuyển thứ tự đối tượng lên 1 bậc |
| moveDown() | void | Di chuyển thứ tự đối tượng xuống 1 bậc |
| niceName() | String | Lấy thông tin cơ bản về đối tượng |
| prevObj() | T | Trả về đối tượng có thứ tự trước 1 bậc |
| nextObj() | T | Trả về đối tượng có thứ tự sau 1 bậc |

2. \_EFEventRegisterInterface

- Mức truy cập: public.

- Loại: giao diện.

- Mô tả: các lớp thực thể muốn đăng ký gọi ngược (callback) bởi các sự kiện khi EF làm việc sẽ thực thi lớp giao diện này.

Bảng PL2.2: Thiết kế giao diện \_EFEventRegisterInterface

|  |  |  |  |
| --- | --- | --- | --- |
| Tên phương thức | Kiểu tham số | Kiểu trả về | Mô tả |
| onBeforeAdded |  | void | Được gọi trước khi thêm vào CSDL |
| onAfterAdded |  | void | Được gọi sau khi thêm vào CSDL |
| onBeforeUpdated |  | void | Được gọi trước khi cập nhật xuống CSDL |
| onAfterUpdated |  | void | Được gọi sau khi cập nhật xuống CSDL |
| onBeforeDeleted |  | void | Được gọi trước khi xóa khỏi CSDL |
| \*Không có sự kiện "onAfterDeleted" vì khi đối tượng đã bị loại khỏi hệ giám sát của EF thì sẽ không khả dụng để gọi ngược | | | |

3. \_EntityAbstract1<T>

- Mức truy cập: public.

- Loại: lớp ảo.

- Lớp cha: không có.

- Lớp giao diện thực thi: \_EFEventRegisterInterface, \_CRUDInterface<T>.

- Mô tả: các lớp thực thể bắt buộc phải có kế thừa trực tiếp hoặc gián tiếp lớp này. Vì đây là các thuộc tính cơ bản cần phải có.

- Thuộc tính và phương thức

Bảng PL2.3: Thiết kế lớp ảo \_EntityAbstract1<T>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mức  truy  cập | Loại  truy  cập | Tên | Kiểu dữ liệu | Mô tả |
| public |  | id | Guid | Khóa chính |
| public |  | subId | string | Mã phụ |
| public |  | date\_create | DateTime? | Ngày tạo |
| public |  | date\_modified | DateTime? | Ngày cập nhật gần nhất |
| public |  | mota | String | Mô tả |
| public |  | order | long? | Dùng để sắp xếp thứ tự |
| protected | static | db | OurDBContext | Truy xuất đến lớp cung cấp dữ liệu |
| public | static | USNAME | String | Tên tiếng anh của lớp này |
| public | static | VNNAME | String | Tên tiếng việt của lớp này |
| protected |  | init() | void | Hàm khởi tạo giá trị ban đầu khi đối tượng được tạo mới |
| protected | static | calculate  OrderColumn() | void | Tự động tính toán trường order cho lớp thực thể T |
| public | static | convert() | List<T> | Chuyển danh sách Id thành danh sách đối tượng |
| protected |  | buildLog() | Dictionary  <string, string> | Xây dựng thông tin đối tượng khi lưu vết hệ thống |

4. \_EntityAbstract2<T>

- Mức truy cập: public.

- Loại: lớp ảo.

- Lớp cha: \_EntityAbstract1<T>.

- Lớp giao diện thực thi: kế thừa từ lớp cha.

- Mô tả: các lớp có mối quan hệ n - n với hình ảnh sẽ kế thừa lớp này.

- Thuộc tính và phương thức

Bảng PL2.4: Thiết kế lớp ảo \_EntityAbstract2<T>

|  |  |  |  |
| --- | --- | --- | --- |
| Mức  truy  cập | Tên | Kiểu dữ liệu | Mô tả |
| public | hinhanhs | ICollection<HinhAnh> |  |

5. \_EntityAbstract3<T>

- Mức truy cập: public.

- Loại: lớp ảo.

- Lớp cha: \_EntityAbstract1<T>.

- Lớp giao diện thực thi: kế thừa từ lớp cha.

- Mô tả: các lớp cần xác thực đăng nhập sẽ kế thừa lớp này.

- Thuộc tính và phương thức.

Bảng PL2.5: Thiết kế lớp ảo \_EntityAbstract3<T>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mức  truy  cập | Loại truy cập | Tên | Kiểu dữ liệu | Mô tả |
| public |  | hoten | String | Họ tên đầy đủ |
| public |  | username | String | Tên đăng nhập |
| public |  | password | String | Mật khẩu dạng băm |
| public | static | checkLoginById() | Boolean | Kiểm tra đăng nhập bằng Id |
| public | static | checkLoginByUserName() | Boolean | Kiểm tra đăng nhập bằng tên đăng nhập |
| public | static | isUsernameExist() | Boolean | Kiểm tra tên đăng nhập có tồn tại |
| public | static | getByUserName() | T | Lấy đối tượng bằng tên đăng nhập |
| public |  | changePassword() | int | Đổi mật khẩu |
| public | static | hashPassword() | String | Hàm băm mật khẩu |

PHỤ LỤC C

- Do số lượng lớp nhiều, không thể trình bày hết, nên chỉ trình bày đại diện một vài lớp.

1. CoSo

- Mức truy cập: public.

- Loại: lớp cứng.

- Lớp cha: \_EntityAbstract2<CoSo>.

- Lớp giao diện thực thi: kế thừa từ lớp cha.

- Mô tả: cơ sở.

- Thuộc tính và phương thức.

Bảng PL3.1: Thiết kế lớp cứng CoSo

|  |  |  |  |
| --- | --- | --- | --- |
| Mức  truy  cập | Tên | Kiểu dữ liệu | Mô tả |
| public | ten | String | Tên cơ sở |
| public | diachi | String | Địa chỉ |
| public | days | ICollection<Dayy> |  |
| public | vitris | ICollection<ViTri> |  |
| public | permissions | ICollection<Permission> |  |

2. ViTri

- Mức truy cập: public.

- Loại: lớp cứng.

- Lớp cha: \_EntityAbstract1<ViTri>.

- Lớp giao diện thực thi: kế thừa từ lớp cha.

- Mô tả: vị trí.

- Thuộc tính và phương thức.

Bảng PL3.2: Thiết kế lớp cứng ViTri

|  |  |  |  |
| --- | --- | --- | --- |
| Mức  truy  cập | Tên | Kiểu dữ liệu | Mô tả |
| public | coso | CoSo |  |
| public | day | Dayy |  |
| public | tang | Tang |  |
| public | phongs | ICollection<Phong> |  |
| public | request() | ViTri | Lấy đối tượng ViTri theo cơ sở, dãy hoặc tầng |

3. Phong

- Mức truy cập: public.

- Loại: lớp cứng.

- Lớp cha: \_EntityAbstract2<Phong>.

- Lớp giao diện thực thi: kế thừa từ lớp cha.

- Mô tả: phòng (phòng học, phòng chức năng).

- Thuộc tính và phương thức.

Bảng PL3.3: Thiết kế lớp cứng Phong

|  |  |  |  |
| --- | --- | --- | --- |
| Mức  truy  cập | Tên | Kiểu dữ liệu | Mô tả |
| public | ten | String | Tên phòng |
| public | nhanvienpt | NhanVienPT | Nhân viên phụ trách được phân công mở cửa phòng |
| public | quantrivien | QuanTriVien | Quản trị viên được phân công quản lý (khác với phân quyền) |
| public | vitri | ViTri |  |
| public | permissions | ICollection<Permission> |  |
| public | ctthietbis | ICollection<CTThietBi> |  |
| public | logthietbis | ICollection<LogThietBi> |  |
| public | sucophongs | ICollection<SuCoPhong> |  |

4. ThietBi

- Mức truy cập: public.

- Loại: lớp cứng.

- Lớp cha: \_EntityAbstract2<ThietBi>.

- Lớp giao diện thực thi: kế thừa từ lớp cha.

- Mô tả: thiết bị.

- Thuộc tính và phương thức.

Bảng PL3.4: Thiết kế lớp cứng ThietBi

|  |  |  |  |
| --- | --- | --- | --- |
| Mức  truy  cập | Tên | Kiểu dữ liệu | Mô tả |
| public | ten | String | Tên thiết bị |
| public | loaithietbi | LoaiThietBi | Thuộc loại thiết bị |
| public | ngaymua | DateTime? | Ngày mua |
| public | ctthietbis | ICollection<CTThietBi> |  |
| public | logthietbis | ICollection<LogThietBi> |  |

5. CTThietBi

- Mức truy cập: public.

- Loại: lớp cứng.

- Lớp cha: \_EntityAbstract2<CTThietBi>.

- Lớp giao diện thực thi: kế thừa từ lớp cha.

- Mô tả: ghép thiết bị với phòng.

- Thuộc tính và phương thức.

Bảng PL3.5: Thiết kế lớp cứng CTThietBi

|  |  |  |  |
| --- | --- | --- | --- |
| Mức  truy  cập | Tên | Kiểu dữ liệu | Mô tả |
| public | soluong | int | Số lượng |
| public | phong | Phong | Thuộc phòng |
| public | thietbi | ThietBi | Thuộc thiết bị |
| public | tinhtrang | TinhTrang | Thuộc tình trạng |
| public | ngay | DateTime? | Ngày lắp vào phòng |
| public | dichuyen() | int | Di chuyển thiết bị sang phòng khác hoặc đổi tình trạng thiết bị |
| private | writelog() | int | Lưu vết thay đổi |